REPORT DOCUMENTATION PAGE

DTIC FILE COPY

Form Approved OMB No. 0704-0188

(2)

...ted to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, ...reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this ...burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson ...Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

AD-A228 797

...RT DATE	3. REPORT TYPE AND DATES COVERED Final Report 1 Jun 86 TO 29 Nov 89
Research in Optical Symbolic Tasks	5. FUNDING NUMBERS 2305/B1
6. AUTHOR(S) Professor Keith Jenkins	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Southern California Signal & Image Processing Unit Los Angeles, CA 90089-0272	8. PERFORMING ORGANIZATION REPORT NUMBER TR-90 1180
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NE Bldg 410 BOLLING AFB WASHINGTON DC 20332-6448 Dr Alan E. Craig	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFOSR-86-0196
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED	12b. DISTRIBUTION CODE

DTIC ELECTE NOV 16 1990 S B D

13. ABSTRACT (Maximum 200 words)

The research findings of the AFOSR Grant AFOSR-86-0196, "Optical Symbolic Computing Tasks" are summarized. The grant period was 1 June 1986 - 29 November 1989. Specifically, we have concentrated on the following topics: complexity studies for optical neural and digital systems, architecture and models for optical computing, learning algorithms for neural networks and applications of neural networks for early vision problems such as image restoration, texture segmentation, computation of optical flow and stero. A number of conference and journal papers reporting the research findings have been published. A list of publications and presentation is given at the end of the report along with a set or reprints. (KR)

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL (UNLIMITED)

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18

FINAL TECHNICAL REPORT

RESEARCH IN OPTICAL SYMBOLIC COMPUTING TASKS

Grant AFOSR-86-0196

Research Period 1 June 1986 - 29 November 1989

Contents

1 Complexity of Optical Neural and Digital Systems 4
1.1 Digital Optical Parallel System Complexity 4
1.2 Connectivity and Hierarchical Neural Networks 4

2 Architecture and Models for Computing 6
2.1 Computation Models and Digital Shared Memory Architectures 6
2.2 Incoherent Optical Neuron 6

3 Learning Algorithms 7
3.1 Potential Difference Learning 7
3.2 Stochastic Learning Networks for Computer Vision 7

4 List of Publications 10

5 List of Oral Presentations 12

6 Copies of reprints resulting in whole or in part from this grant 14

Abstract

The research findings of the AFOSR Grant AFOSR-86-0196, "Optical Symbolic Computing Tasks" are summarized. The grant period was 1 June 1986 - 29 November 1989. Specifically, we have concentrated on the following topics: complexity studies for optical neural and digital systems, architecture and models for optical computing, learning algorithms for neural networks and applications of neural networks for early vision problems such as image restoration, texture segmentation, computation of optical flow and stereo. A number of conference and journal papers reporting the research findings have been published. A list of publications and presentations is given at the end of the report along with a set of reprints.

1 Complexity of Optical Neural and Digital Systems

1.1 Digital Optical Parallel System Complexity

Our study of digital optical system complexity has included a comparison of optical and electronic interconnection network complexity, and a study of design and complexity tradeoffs for the implementation of a shared memory parallel computer. The complexity of some common interconnection networks have been analyzed for optical and electronic VLSI implementations in detail. The optical system used for analysis was the hybrid 2-hologram interconnection system of Jenkins, et al. Area complexity was compared and found to be

	VLSI	**OPTICS**
Banyan	$0(n^2)$	$0(nlog^2n)$
Shuffle/Exchange	$0(n^2/logn)$	$0(n^2logn)$
Hypercube	$0(n^2)$	$0(nlog^2n)$
2-D Cellular Hypercube	$\geq 0(n^2)$	$0(n)$

It should be noted that the electronic results have received a great deal of work on using variou clever tricks and algorithms to reduce the result to near optimum. The optics case was only investigated by us and can likely be reduced further by using different layouts. The Banyan and shuffle/exchange networks are isomorphic and for them, optics has lower complexity for large n. An example of how the optical complexity can be lowered can be seen in the hypercube network. The 2-D cellular hypercube is identical to two overlapping hypercube networks, so it has twice as many interconnections, yet its optical area complexity is much lower because it is space-invariant.

We have more recently applied our expertise and results in complexity analysis to the use of optics in the implementation of a parallel digital shared memory computer. This is described in Section 2.1.

1.2 Connectivity and Hierarchical Neural Networks

In neural networks the connectivity can be very high and in many cases the nets are even fully connected. As has been shown by Psaltis, even optical systems may not be able to provide this much connectivity for nets with large numbers of neuron units (i.e., 2-D arrays of neuron units). One technique for optically reducing the physical interconnection requirements is to take advantage of any symmetry or regularity in an interconnection. Since neural nets are particularly useful for random problems, and this may imply random interconnections, at first thought utilizing symmetry may not seem plausible. In the case of vision, however, there is typically a high degree of regularity or symmetry to the interconnection. In addition, even for other applications, many nets may have a hierarchical structure, and this can often imply repetition in the interconnections. For example, a network that utilizes a number representation scheme with binary neurons may have the same interconnections repeated for each group of neurons that represents a number. While number representation techniques that have been described for neural networks do not have this repetition, we have found that variants of them do. We have designed such network structures that have repeated blocks, one for each represented number, and have incorporated proper update rules for the neurons to ensure convergence of the net. This work has focused on single layer feedback networks used for combinatorial optimization. This yields a hierarchical network in the sense that each block represents the lower level, and the interconnections from block to block represent the higher level. It may also be extendable to hierarchies with more than two levels.

2 Architecture and Models for Computing

2.1 Computation Models and Digital Shared Memory Architectures

We have investigated abstract models of parallel computation that have been invented and studied fairly extensively by the computer science community. These models are: (1) inherently parallel, (2) abstract and thereby divorced from the constraints of any technology, and (3) more powerful than physically realizable parallel computers. We have found that these models have substantial implications in the design of optical or hybrid optical/electronic parallel digital computers. Using these models (instead of parallel electronic computer architectures) as a starting point in the design of parallel optical machines can potentially yield novel and more powerful architectures that can take better advantage of the characteristics of optical hardware. It is interesting to note that the primary aspects of these models that make them impossible to implement physically are the extremely parallel and powerful interconnection network, and the completely parallel and contention-free access to shared memory; both of these aspects are potential advantages that optical systems have over electronic systems.

We have proceeded, as suggested above, to use these models to develop an initial design for a new architecture for parallel optical/electronic computing. It is based on shared memory, an optical interconnection network, and electronic processing. It can potentially use optics in conjunction with appropriate control and routing techniques to achieve functionality and processing power heretofore unachieved with optical systems. In addition, it addresses the issue of what kinds of parallel access shared memories may be desirable, and how they might be addressed and incorporated into an architecture. It is anticipated that this work will continue after the end of this grant.

2.2 Incoherent Optical Neuron

A general neuron unit, used in a neural network, must incorporate both positive and negative (excitatory and inhibitory) inputs. We have invented, modeled, simulated, and experimentally demonstrated a new method for incorporating both types of inputs in an optical neuron unit that uses only incoherent optical devices. This incoherent optical neuron (ION) is cascadable, can be used efficiently in partially as well as fully connected neural networks, and can be used with either coherent or incoherent optical interconnections. This work was also supported in part by AFOSR under the University Research Initiative program.

3 Learning Algorithms

3.1 Potential Difference Learning

We developed a new learning algorithm, *potential difference learning.* It is based on a temporal difference of the neuron unit potential,

$$\Delta w_{ij} \propto \Delta p_i x_j$$

where Δw_{ij} is the weight increment from neuron j to neuron i, Δp_i is the temporal difference of potential, and x_j is the jth input to neuron i, for self-organization in neural networks. Depending on the time sequence of the input patterns during learning, it can learn based on the input patterns themselves or based on the time difference of input patterns. It has no weight overflow as with a strict Hebbian law. It can, with suitable presentation of the input patterns, also be used to *unlearn* or *erase* stored states in an associative memory without access to the individual weights and without reversing the sign of the learning gain constant.

We have simulated potential difference learning on two different networks: (1) an Amari network, i.e. a single layer fully connected network with feedback, used as an associative memory, and (2) a 3-layer network used as two associative memories with a hidden layer to relate pairs of stored vectors. These are described in a paper attached to this report.

3.2 Stochastic Learning Networks for Computer Vision

We have developed stochastic learning networks for an important problem in Computer Vision, viz, texture segmentation. Our approach is based on minimizing an energy function, derived through the representation of textures as Markov Random Fields (MRF). We use the Gauss Markov Random Field (GMRF) to represent the texture intensities and an Ising model to characterize the label distribution. We first used an adaptive Cohen-Grossberg/Hopfield network to minimize the resulting energy function. The solution obtained is a local optimum in general and may not be satisfactory in many cases. Although stochastic algorithms like simulated annealing have a potential of finding a global optimum, they are computationally expensive. We have developed an alternate approach based on the theory of *learning automaton* which introduces *stochastic learning* into the iterations of the Hopfield network. This approach consists of a two stage process with learning and relaxation alternating with each other and because of its stochastic nature has the potential of escaping the local minima.

The learning part of the system consists of a team of automata A_s, one automaton for each pixel site. Each automaton A_s at site s maintains a time varying probability vector $\mathbf{P_s} = [p_{s1} ..., p_{sL}]$ where p_{sk} is the probability of assigning the texture class k to the pixel site s. Initially all these probabilities are equal. At the beginning of each cycle the learning system will choose a label configuration based on this probability distribution and present it to the Cohen-Grossberg/Hopfield neural network described above as an initial state. The neural network will then converge to a stable state. The probabilities for the labels in the stable configuration are increased according to the following updating rule: Let k_s be the label selected for the site $s = (i,j)$ in the stable state in the n-th cycle. Let $\lambda(n)$ denote a reinforcement signal received by the learning system in that cycle. Then,

$$p_{s_{k_s}}(n+1) = p_{s_{k_s}}(n) + a\lambda(n)[1 - p_{s_{k_s}}]$$

$$p_{s_j}(n) = p_{s_j}(n)[1 - a\lambda(n)], \forall j \neq k_s$$

for all $s = (i,j), 1 \leq i,j \leq M$.

In the above equation 'a' determines the learning rate of the system. The reinforcement signal determines whether the new state is good compared to the previous one in terms of the energy function. Using the new probabilities, a new initial state is randomly generated for the relaxation network and the process repeats. The above learning rule is called **Linear Reward-Inaction** rule in the learning automata terminology.

We have tested this algorithm in classifying some real textured images. The Hopfield network solution has a misclassification error of about 14% without learning. The error decreased to 6.8% when stochastic learning was introduced. When simulated annealing was tried the error rate is 6.3%, but the number of iterations were considerably more. In general stochastic algorithms seem to perform better than any deterministic scheme.

Recently, we have extended our approach to the unsupervised case. In this method, the image is divided into a number of non-overlapping regions and the GMRF parameters are computed from each of these regions. A simple clustering scheme is used to merge these regions. The parameters of the model estimated from the clustered segments are then used in the deterministic and stochastic algorithms mentioned earlier. Details of the unsupervised texture segmentation results are very encouraging.

Under partial support from this grant and the USC URI Center for the Integration of Optical Computing, we have developed an adaptive neural network (NN) based algorithm for a fundamental problem in image processing, viz., restoration of a blurred and noise corrupted image. In this method, we have used a NN model to represent a possibly nonstationary image whose gray level function is the simple sum of neuron state variables. The restoration procedure consists of two stages: estimation of the parameters of the NN model and reconstruction of images. During the first stage, the parameters are estimated by comparing the energy function of the network to a constrained error function. the nonlinear restoration method is then carried out iteratively in the second stage by using a dynamic algorithm to minimize the energy function. We have also developed a practical algorithm with reduced computational complexity. Comparisons to other methods such as the Singular Value Decomposition (SVD) pseudoinverse filter, minimum mean square error filter, and modified Minimum Mean Square Error (MMSE) filter using the GMRF models showed that the NN based method performed the best.

We have extended these techniques for other vision problems such as computation of optical flow, static and motion stereo. The network for computation of optical flow from two or more image frames uses rotation invariant features known as principal curvatures. A set of layers of binary neurons is used to represent the flow field. Each neuron receives all inputs from itself and other neurons in a local neighborhood in the same layer. Computation of optical flow is carried out by neuron evaluation using a parallel updating scheme. Using information regarding the occluding elements, the network automatically locates motion discontinuities. To improve the accuracy of the estimated flow field, two algorithms, batch and recursive using multiple frames are presented. The batch algorithm integrates information from all images simultaneously by embedding them into bias inputs of the network, while the recursive algorithm uses a procedure to update the bias inputs of the network. Satisfactory results have been obtained using these methods on a number of real images.

The network for static and motion stereo uses a similar approach using first derivatives estimated by filtering discrete orthogonal polynomials. The recursive algorithm for motion stereo takes into account various situations such as split motion, fusion motion in obtaining updated values of disparity.

4 List of Publications

1. Y.T. Zhou, R. Chellappa and B.K. Jenkins, "A Novel Approach to Image Restoration Based on a Neural Network," *Proc. IEEE First International Conference on Neural Networks,* San Diego, Vol. IV, pp. 269-276 June 1987.

2. C.H. Wang and B.K. Jenkins "Potential Difference Learning and Its Optical Architecture," accepted for *Proc. Soc. Photo-Opt. Instr. Eng.,* Vol. 882, January 1988.

3. Y.T. Zhou, R. Chellappa, A. Vaid, and B.K. Jenkins, "Image Restoration Using a Neural Network," IEEE Trans. Acoust. Speech and Signal Processing, Vol. ASSP-36, pp. 1141-1151, July 1988.

4. C.H. Wang and B.K. Jenkins, "The Implementation Consideration of a Subtracting Incoherent Optical Neuron," *The IEEE International Conference on Neural Networks,* San Diego, July 1988.

5. B.K. Jenkins and C.L. Giles, "Superposition in Optical Computing," accepted for *Proc. ICO Topical Meeting on Optical Computing,* Toulon, France, to appear, August 1988.

6. B.S. Manjunath and R. Chellappa, "Stochastic Learning Networks for Texture Segmentation", (Accepted for Publication at the Twenty Second Annual Asilomar Conference on Signals, Systems and Computers), Pacific Grove, CA, October 1988.

7. B.K. Jenkins and C.H. Wang, "Model for an Incoherent Optical Neuron that Subtracts," *Optics Letters,* Vol. 13, pp. 892-894, October 1988.

8. C.H. Wang and B.K. Jenkins, "Implementation of a Subtracting Incoherent Optical Neuron," *Proc. IEEE Third Annual Parallel Processing Symposium,* Fullerton, CA, March 1989.

9. C.H. Wang and B.K. Jenkins, "Subtracting Incoherent Optical Neuron Model: Analysis, Experiment, and Applications," *Applied Optics,* Vol. 29, pp. 2171-2186, May 10, 1990.

10. C.L. Giles and B.K. Jenkins, "Complexity Implications of Optical Parallel Computing," *Proc. of the Twentieth Annual Asilomar Conference on Signals, Systems, and Computers,*

Pacific Grove, CA, November 1-12, 1986.

11. B.S. Manjunath, T. Simchony and R. Chellappa, "Stochastic and Deterministic Networks for Texture Segmentation," *IEEE Trans. Acoust., Speech and Signal Processing,* Vol. ASSP-38, pp. 1039-1049, June 1990.

12. B.S. Manjunath and R. Chellappa, "A Note on Unsupervised Texture Segmentation," *Proc. Int'l. Conf. on Acoust., Speech, and Signal Proc.,* New York, NY, April 1988.

13. Y.T. Zhou and R. Chellappa, "Neural Network Algorithms for Motion Stereo," *Int'l. Joint Conf. on Neural Networks,* Washington, D.C., Vol. III, pp 251-258, June 1989.

14. Y.T. Zhou and R. Chellappa, "A Network for Motion Perception," *Int'l. Joint Conf. on Neural Networks,* San Diego, June 1990.

5 List of Oral Presentations

1. B.K. Jenkins, "Optical Computing: Status and Prospects," briefing given to Dr. Bernard Paiewonsky, Deputy for Advanced Technology, Air Force, The Pentagon, Washington, D.C., September 1987.

2. B.K. Jenkins and C.H. Wang, "Model for An Incoherent Optical Neuron that Subtracts," annual meeting of the Optical Society of America, paper PD4, Rochester, New York, October 1987.

3. B.K. Jenkins and C. Lee Giles, "Massively Parallel Optical Computing," IEEE Communications Theory Workshop, Sedonna, Arizona, April 18-21, 1988.

4. B.K. Jenkins, "Optical Computing: Status and Prospects," IEEE Orange County Chapter meeting, Santa Ana, California, May 1988.

5. B.K. Jenkins, and C.L. Giles, "Optical Architecture Implications of Parallel Computation Models," Annual Meeting of the Optical Society of America, Seattle, WA, paper ML4, October 1986.

6. C.L. Giles and B.K. Jenkins, "Parallel Computation Models for Optical Computing," Annual Meeting of the Optical Society of America, Seattle, WA, paper ML1, October 1986.

7. B.K. Jenkins and C.L. Giles, "Parallel Computation and Optics," invited paper, Gordon Conference on Holography and Optical Information Processing, Santa Barbara, CA, January 1987.

8. B.K. Jenkins, "Roles of Optics in Digital Computing," invited presentation, Meeting of the IEEE Computer Society, Orange County Chapter, January 1987.

9. B.K. Jenkins, "Computer Elements and Optical Computing," IEEE Computer Elements Workshop, Vail, CO, June 27-29, 1988.

10. B.K. Jenkins, "Implementation Issues in Optical Neural Networks," invited paper, *Proc. Twenty-Second Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific

Grove, CA, November 1988.

11. B.K. Jenkins, "Optical Neural Nets for Computer Vision, Pattern Recognition, and Associative Memory," invited paper, *Florida Annual Bio Technology Symposium,* February 6-10, 1989.

12. B.K. Jenkins, "Free-Space Optical Systems for Communications, Interconnections, and Computing," invited paper, *Topical Meeting on Photonic Switching,* Optical Society of America, Salt Lake City, UT, March 1-3, 1989.

13. B.K. Jenkins, "Parallel Computing Using Optics," invited paper, *NSF - Industry - University Symposium on National Agenda and Priorities in Computer Science and Engineering Research,* sponsored by NSF and IBM, Yorktown Heights, NY, April 24-26, 1989.

14. B.K. Jenkins, "Implementation Issues for Optical Neural Networks," Osaka University, Osaka, Japan, July 11, 1989; also Electro-Technical Laboratory, Tsukuba, Japan, July 5, 1989; and NTT Corporation, Yokosuka, Japan, July 7, 1989.

15. C.H. Wang and B.K. Jenkins, "Subtracting Incoherent Optical Neuron: Experimental Demonstration," *Annual Meeting,* Optical Society of America, Orlando, FL, October 15-20, 1989.

16. Paper 1, 6, 12-15 were presented at the respective conferences.

6 Copies of reprints resulting in whole or in part from this grant

A Novel Approach to Image Restoration Based on a Neural Network[1]

Y. T. Zhou, R. Chellappa and B. K. Jenkins

Signal and Image Processing Institute
Department of EE-Systems
University of Southern California

Abstract

A novel approach for restoration of gray level images degraded by a known shift invariant blur function and additive noise is presented using a neural computational model. A neural network model is employed to represent an image whose gray level function is the simple sum of the neuron state variables. The restoration procedure consists of two stages: estimation of the parameters of the neural network model and reconstruction of images. During the first stage, image noise is suppressed and the parameters are estimated. The restoration is then carried out iteratively in the second stage by using a dynamic algorithm to minimize the energy function of an appropriate neural network. Owing to the model's fault-tolerant nature and computation capability, a high quality image is obtained using this approach. A practical algorithm with reduced computational complexity is also presented. Several computer simulation examples involving synthetic and real images are given to illustrate the usefulness of our method.

1 Introduction

Image restoration is an important problem in early vision processing to recover an ideal high quality image from a degraded recording. Restoration techniques are applied to remove (1) system degradations such as blur due to optical system aberrations, atmospheric turbulence, motion and diffraction; and (2) statistical degradations due to noise. Over the last 20 years, various methods such as the inverse filter, Wiener filter, Kalman filter, SVD pseudoinverse and many other model based approaches, have been proposed for image restoration. One of the major drawbacks of most of the image restoration algorithms is the computational complexity, so much so that many simplifing assumptions have been made to obtain computationally feasible algorithms. An artificial neural network system that can perform extremely rapid parallel computation seems to be very attractive for image processing applications; preliminary investigations to various problems such as pattern recognition and image processing are very promising [1].

In this paper, we use a neural network model containing redundant neurons to restore gray level images degraded by a known shift invariant blur function and noise. It is based on the model described in [2] [3] using a simple sum number representation [4]. The image gray levels are represented by the simple sum of the neuron state variables which take binary values of 1 or 0. The observed image is degraded by a shift-invariant function and noise. The restoration procedure consists of two stages: estimation of the parameters of the neural network model, and reconstruction of images. During the first stage, the image noise is suppressed and the parameters are estimated. The restoration is then carried out by using a dynamic iterative algorithm to minimize the energy function of the neural network. Owing to the model's fault-tolerant nature and computation capability, a high quality image is obtained using our approach. We illustrate the usefulness of this approach by using both synthetic and real images degraded by a known shift-invariant blur function with or without noise.

[1]This research work is partially supported by the AFOSR Contract No. F-49620-87-C-0007.

2 Image Representation Using a Neural Network

We use a neural network containing redundant neurons for representing the image gray levels. The model consists of $L^2 \times M$ mutually interconnected neurons, where L is the size of image and M is the maximum value of the gray level function. Let $V = \{v_{i,k}, 1 \leq i \leq L^2, 1 \leq k \leq M\}$ be a binary state set of the neural network with $v_{i,k}$ (1 for firing and 0 for resting) denoting the state of the (i,k)th neuron. Let $T_{i,k;j,l}$ denote the strength (possibly negative) of the interconnection between neuron (i,k) and neuron (j,l). We require symmetry

$$T_{i,k;j,l} = T_{j,l;i,k} \quad for \quad 1 \leq i,j \leq L^2 \; and \; 1 \leq l,k \leq M$$

We also insist that the neurons have self-feedback, i.e. $T_{i,k;i,k} \neq 0$. In this model, each neuron (i,k) randomly and asynchronously receives inputs $\sum T_{i,k;j,l} v_{j,l}$ from all neurons and a bias input $I_{i,k}$

$$u_{i,k} = \sum_{j}^{L^2} \sum_{l}^{M} T_{i,k;j,l} v_{j,l} + I_{i,k} \tag{1}$$

Each $u_{i,k}$ is fed back to corresponding neurons after thresholding

$$v_{i,k} = g(u_{i,k}) \tag{2}$$

where $g(x)$ is a nonlinear function whose form can be taken as

$$g(x) = \begin{cases} 1 & if \;\; x \geq 0 \\ 0 & if \;\; x < 0. \end{cases} \tag{3}$$

In this model, the state of each neuron is updated by using the latest information about other neurons.

The image is described by a finite set of gray level functions $\{x(i,j), 1 \leq i,j \leq L\}$ with $x(i,j)$ (positive integer number) denoting the gray level of the cell (i,j). The image gray level function can be represented by a simple sum of the neuron state variables as

$$x(i,j) = \sum_{k=1}^{M} v_{m,k} \tag{4}$$

where $m = i \times L + j$. Here the gray level functions have degenerate representations. Use of this redundant number representation scheme yields advantages such as fault-tolerance and convergence to the solution [4].

If we scan the 2-D image by rows and stack them as a long vector, then the degraded image vector can be written as

$$\underline{Y} = H\underline{X} + \underline{N} \tag{5}$$

where H is the $L^2 \times L^2$ point spread function (or blur) matrix, and $\underline{X}$, $\underline{Y}$ and $\underline{N}$ are the $L^2 \times 1$ long original, degraded and noise vectors, respectively. This is similar to the simultaneous equations solution of [4], but differs in that (5) includes a noise term.

The shift-invariant blur function can be written as a convolution over a small window, for instance, it takes the form

$$h(k,l) = \begin{cases} \frac{1}{2} & if \;\; k = 0, \; l = 0 \\ \frac{1}{16} & if \;\; |k|, \; |l| \leq 1, \;\; (k,l) \neq (0,0) \end{cases} \tag{6}$$

accordingly, the "blur matrix" H will be a block Toeplitz or block circulant matrix (if the image has periodic boundaries).

3 Estimation of Model Parameters

The neural model parameters, the interconnection strengths and the bias inputs, can be determined in terms of the energy function of the neural network. As defined in [2], the energy function of the neural network can be written as

$$E = -\frac{1}{2}\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M} T_{i,k;j,l}\, v_{i,k}\, v_{j,l} + \sum_{i=1}^{L^2}\sum_{k=1}^{M} I_{i,k}\, v_{i,k} \tag{7}$$

In order to use the spontaneous energy–minimization process of the neural network, we reformulate our restoration problem as one of minimizing an energy function defined as

$$E = \frac{1}{2}\|\underline{Y} - H\underline{X}\|^2 \tag{8}$$

where $\|\underline{Z}\|$ is the L_2 norm of $\underline{Z}$. By comparing the terms in the expansion of (8) with the corresponding terms in (7), we can determine the interconnection strengths and the bias inputs as

$$T_{i,k;j,l} = -\sum_{p=1}^{L^2} h_{p,i}\, h_{p,j} \tag{9}$$

and

$$I_{i,k} = \sum_{p=1}^{L^2} y_p\, h_{p,i}. \tag{10}$$

From (9), one can see that the interconnection strengths are determined by the shift–invariant blur function. Hence, $T_{i,k;j,l}$ can be computed without error provided the blur function is known. However, the bias inputs are functions of observation, the degraded image. If the image is degraded by shift–invariant blur function only, then $I_{i,k}$ can be estimated perfectly. Otherwise, the degraded image needs to be preprocessed to suppress the noise if the signal to noise ratio (SNR), defined by

$$SNR = 10\,\log_{10}\frac{\sigma_s^2}{\sigma_n^2} \tag{11}$$

where σ_s^2 and σ_n^2 are variances of signal and noise, respectively, is low.

4 Restoration

Restoration is carried out by the neuron evaluation and image construction procedure. Once the parameters $T_{i,k;j,l}$ and $I_{i,k}$ are obtained using (9) and (10), each neuron can randomly and asynchronously evaluate its state and readjust accordingly using (1) and (2). When one quasi–minimum energy point is reached, the image can be constructed by (4).

However, this neural network has self–feedback, i.e. $T_{i,k;i,k} \neq 0$, as a result of a transition the energy function E does not decrease monotonically. This is explained as follows. Define the state change $\Delta v_{i,k}$ of neuron (i,k) and energy change ΔE as

$$\Delta v_{i,k} = v_{i,k}^{new} - v_{i,k}^{old} \quad and \quad \Delta E = E^{new} - E^{old}$$

Consider the energy function

$$E = -\frac{1}{2}\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M} T_{i,k;j,l}\, v_{i,k}\, v_{j,l} - \sum_{i=1}^{L^2}\sum_{k=1}^{M} I_{i,k}\, v_{i,k}, \tag{12}$$

Then the change ΔE due to a change $\Delta v_{i,k}$ is given by

$$\Delta E = -\left(\sum_{j=1}^{L^2} \sum_{l=1}^{M} T_{i,k;j,l}\, v_{j,l} + I_{i,k}\right) \Delta v_{i,k} - \frac{1}{2}\, T_{i,k;i,k}\, (\Delta v_{i,k})^2 \qquad (13)$$

which is not always negative. For instance, if

$$v_{i,k}^{old} = 0, \qquad u_{i,k} = \sum_{j=1}^{L^2} \sum_{l=1}^{M} T_{i,k;j,l}\, v_{j,l} + I_{i,k} > 0,$$

and the threshold function is as in (3), then $v_{i,k}^{new} = 1$ and $\Delta v_{i,k} > 0$. Thus, the first term in (13) is negative. But

$$T_{i,k;i,k} = -\sum_{p=1}^{L^2} h_{p,i}^2 < 0.$$

leading to

$$-\frac{1}{2}\, T_{i,k;i,k}\, (\Delta v_{i,k})^2 > 0.$$

When the first term is less than the second term in (13), then $\Delta E > 0$ (we have observed this in our experiments).

Thus, depending on whether convergence to a local minimal or a global minimal is desired, we can design a deterministic or stochastic decision rule. The deterministic rule is to take a new state $v_{i,k}^{new}$ of neuron (i,k) if the energy change ΔE due to state change $\Delta v_{i,k}$ is less than zero. If ΔE due to state change is > 0, no state change is affected. We have also designed a stochastic rule similar to the one used in simulated annealing techniques [5] [6]. The details of this stochastic scheme are given as follows:

Define a Boltzmann distribution by

$$\frac{p_{new}}{p_{old}} = e^{\frac{-\Delta E}{T}}$$

where p_{new} and p_{old} are the probabilities of the new and old global state, respectively, ΔE is the energy change and T is the parameter which acts like temperature. A new state $v_{i,k}^{new}$ is taken if

$$\frac{p_{new}}{p_{old}} > 1, \qquad or\ if\ \ \frac{p_{new}}{p_{old}} \le 1 \ \ but\ \ \frac{p_{new}}{p_{old}} > \xi$$

where ξ is a random number uniformly distributed in the interval [0,1].

The restoration algorithm can then be summarized as

1. Set the initial state of the neurons.
2. Update the state of all neurons randomly and asynchronously according to the decision rule.
3. Check the energy function; if energy does not change anymore, go to next step; otherwise, go back to step 2.
4. Construct an image using (4).

5 A Practical Algorithm

The algorithm described above is difficult to simulate on a conventional computer due to high computational complexity even for images of reasonable size. For instance, if we have an $L \times L$ image with M gray levels, then L^2M neurons and $\frac{1}{2}L^4M^2$ interconnections are required and L^4M^2 additions and multiplications are needed at each iteration. Therefore, the space and time complexities are $O(L^4M^2)$ and $O(L^4M^2K)$, respectively, where K $O(10)$-$O(100)$ is the number of iterations. When $L = 256$ and

$M = 256$, the space and time complexities will be $O(10^{14})$ and $O(10^{15})$-$O(10^{15})$, respectively. However, simplification is possible if the neurons are sequentially updated .

In order to simplify the algorithm, we begin by reconsidering (1) and (2) of the neural network. Noting that the interconnection strengths given in (9) are independent of subscripts k and l and the bias inputs given in (10) are independent of subscript k, the M neurons used to represent the same image gray level function have the same interconnection strengths and bias inputs. Hence, one set of interconnection strengths and one bias input are sufficient for every gray level function, i.e. the dimensions of the interconnection matrix T and bias input matrix I can be reduced by a factor of M^2. From (1) all inputs received by a neuron, say, the (i,k)th neuron can be written as

$$\begin{aligned} u_{i,k} &= \sum_{j}^{L^2} T_{i,\cdot;j,\cdot} \left(\sum_{l}^{M} v_{j,l}\right) + I_{i,\cdot} \\ &= \sum_{j}^{L^2} T_{i,\cdot;j,\cdot}\, x_j + I_{i,\cdot} \end{aligned} \tag{14}$$

where we have used (4) and x_j is the gray level function of the jth image pixel. Equation (14) suggests that we can use a multivalue number to replace the simple sum number. Since the interconnection strengths are determined by the blur function only as shown in (9), it is easy to see that if the blur function is local, then most interconnection strengths are zeros so that the neurons are locally connected. Therefore, most elements of the interconnection matrix T are zeros. If the blur function is shift invariant taking the form in (6), then the interconnection matrix is block Toeplitz so that only a few elements need to be stored. Based on the value of inputs $u_{i,k}$, the state of the (i,k)th neuron is updated by applying a decision rule. The state change of the (i,k)th neuron in turn causes the gray level function x_i to change

$$x_i^{new} = \begin{cases} x_i^{old} & if\ \Delta v_{i,k} = 0 \\ x_i^{old} + 1 & if\ \Delta v_{i,k} = 1 \\ x_i^{old} - 1 & if\ \Delta v_{i,k} = -1 \end{cases} \tag{15}$$

where $\Delta v_{i,k} = v_{i,k}^{new} - v_{i,k}^{old}$ is the state change of the (i,k)th neuron. The supscripts "new" and "old" are for after and before updating, respectively. We use x_i to respresent the gray level value as well as the output of M neurons representing x_i. Assuming that the neurons of the network are sequentially visited, it is straightforward to prove that the updating procedure can be reformulated as

$$u_{i,k} = \sum_{j}^{L^2} T_{i,\cdot;j,\cdot}\, x_j + I_{i,\cdot} \tag{16}$$

$$\Delta v_{i,k} = g(u_{i,k}) = \begin{cases} \Delta v_{i,k} = 0 & if\ u_{i,k} = 0 \\ \Delta v_{i,k} = 1 & if\ u_{i,k} > 0 \\ \Delta v_{i,k} = -1 & if\ u_{i,k} < 0 \end{cases} \tag{17}$$

$$x_i^{new} = \begin{cases} x_i^{old} + \Delta v_{i,k} & if\ \Delta E < 0 \\ x_i^{old} & if\ \Delta E \geq 0 \end{cases} \tag{18}$$

Note that the stochastic decision rule can also be used in (18). In order to limit the gray level function to the range 0 to 255, after each updating step we have to check the value of the gray level function x_i^{new}. Equations (16), (17) and (18) give a much simpler algorithm. This algorithm is summarized below:

1. Take the degraded image as the initial value.

2. Sequentially visit all numbers (image pixels). For each number, use (16), (17) and (18) to update it repeatedly until no further change, i.e. if $\Delta v_{i,k} = 0$ or energy change $\Delta E \geq 0$, then move to next one.

3. Check the energy function; if energy does not change anymore, a restored image is obtained;

otherwise, go back to step 2 for another iteration.

The calculations of the inputs $u_{i,k}$ of the (i,k)th neuron and the energy change ΔE can be simplified furthermore. When we update the same image gray level function repeatedly, the inputs received by the current neuron (i,k) can be computed by making use of the previous result

$$u_{i,k} = u_{i,k-1} + \Delta v_{i,k}\, T_{i,\cdot;i,\cdot} \tag{19}$$

where $u_{i,k-1}$ is the inputs received by the $(i,k-1)$th neuron. The energy change ΔE due to the state change of the (i,k)th neuron can be calculated as

$$\Delta E = -u_{i,k}\, \Delta v_{i,k} - \frac{1}{2}\, T_{i,\cdot;i,\cdot}\, (\Delta v_{i,k})^2 \tag{20}$$

If the blur function is shift invariant, all these simplifications reduce the space and time complexities significantly from $O(L^4M^2)$ and $O(L^4M^2K)$ to $O(L^2)$ and $O(ML^2K)$, respectively. Since every gray level function needs only a few updating steps after the first iteration, the computation at each iteration is $O(L^2)$. The resulting algorithm can be easily simulated on mini–computers for images, as large 512×512.

6 Computer Simulations

The practical algorithm described in the previous section was applied to the synthetic and real images on a Sun-3/160 Workstation. In all cases, only the deterministic decision rule was used. The results are summarized in Figure 1 and 2.

Figure 1 shows the results for the synthetic image. The original image shown in Figure 1(a) is of size 32×32 with 3 gray levels. The image was degraded by convolving with a 3×3 blur function as in (6) using a circulant boundary condition; 22 dB white Gaussian noise was added after convolution. A perfect image was obtained after 6 iterations without preprocessing. We set the state of all neurons to equal 1, i.e. firing as initial condition.

Figure 2(a) shows the original girl image. The original image is of size 256×256 with 256 gray levels. The variance of the original image is 2826.128. It was degraded by a 5×5 uniform blur function. A small amount of quantization noise was introduced by quantizing the convolution results to 8 bits. The noisy blurred image is shown in Figure 2(b). For comparison purpose, Figure 2(c) shows the output of an inverse filter [7], completely overridden by the amplified noise and the ringing effects due to the ill conditioned of the blur matrix H. Since the blur matrix H corresponding to the 5×5 uniform blur function is not singular, the pseudoinverse filter [7] and the inverse filter have the same output. The restored image by using our approach is shown in Figure 2(d). In order to eliminate the ringing effect, due to the boundary conditions, we took the 4 pixel wide boundaries from the original image and updated the interior region (248×248) of the image only. The blurred imgage was used as an initial condition for accelerating the convergence. The total number of iterations was 213 (when the energy function did not change anymore). The square error (i.e. energy function) defined in (8) is 0.02543 and the square error between the original and restored imges is 66.5027.

7 Conclusion

This paper has introduced a novel approach to restore gray level images degraded by a shift invariant blur function and additive noise. The restoration procedure consists of two steps: parameter estimation and image reconstruction. In order to reduce the computational complexity, a practical algorithm which is equivalent to the original one is developed under the assumption that the neurons are sequentially visited. The image is generated iteratively by updating the neurons representing the image gray levels via a simple sum scheme. As no matrices are inverted, the serious problem of ringing due to the ill conditioned blur matrix H and noise overriding caused by inverse filter or pseudoinverse inverse filter can be avoided. For the case of 2-D uniform blur plus noise, the neural network based approach give high quality images whereas the inverse filter and pseudoinverse filter yield poor results. We see from the experimental results that the error defined by (8) is small while the error between the original image and the restored image is relatively large. This is because the neural network decreases energy according to (8) only. Another reason is that when the blur matrix is singular or near singular, the mapping from $\underline{X}$ to $\underline{Y}$ is not one to one, therefore, the error measure (8) is not reliable anymore. Thus, we have to point out that our approach will not work very well when the bluuring matrix is singular. In our experiments, when the window size of a uniform blur function is 3 x 3, the ringing effect was eliminated by leaving the boundaries of the degraded image without processing. When the window size is 5 x 5, the ringing effect was significantly reduced by using the original image boundaries.

References

[1] N. H. Farhat, D. Psaltis, A. Prata, and E. Paek, "Optical Implementation of the Hopfield Model", *Applied Optics*, vol. 24, No.10, pp. 1469-1475, 15 May 1985.

[2] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, vol. 52, pp. 114-152, 1985.

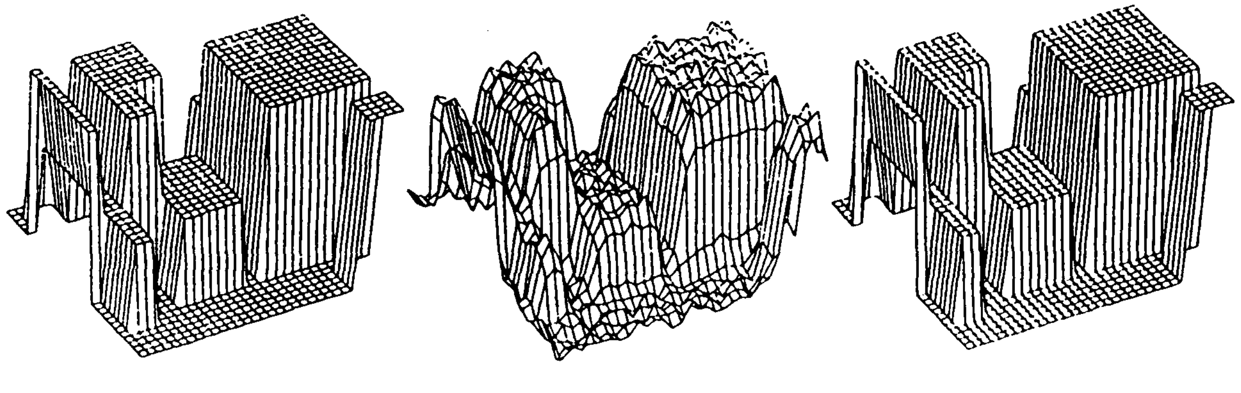
[3] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554-2558, April 1982.

[4] M. Takeda and J. W. Goodman, "Neural Networks for Computation: Number Representations and Programming Complexity", *Applied Optics*, vol. 25, No. 18, pp. 3033-3046, Sept. 1986.

[5] N. Metropolis et al., "Equations of State Calculations by Fast Computing Machines", *J. Chem. Phys.*, vol. 21, pp. 1087-1091, 1953.

[6] S. Kirkpatrick et al., "Optimization by Stimulated Annealing", *Science*, vol. 220, pp. 671-680, 1983.

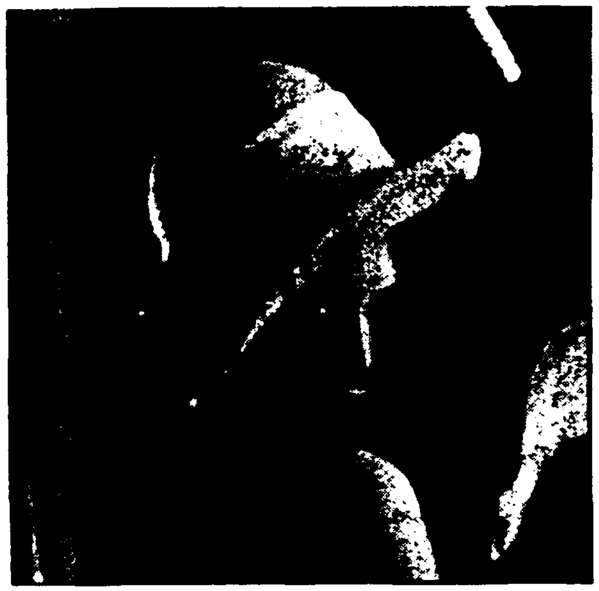
[7] W. K. Pratt, *Digital Image Processing*, John Wiley and Sons, New York, New York, 1978.

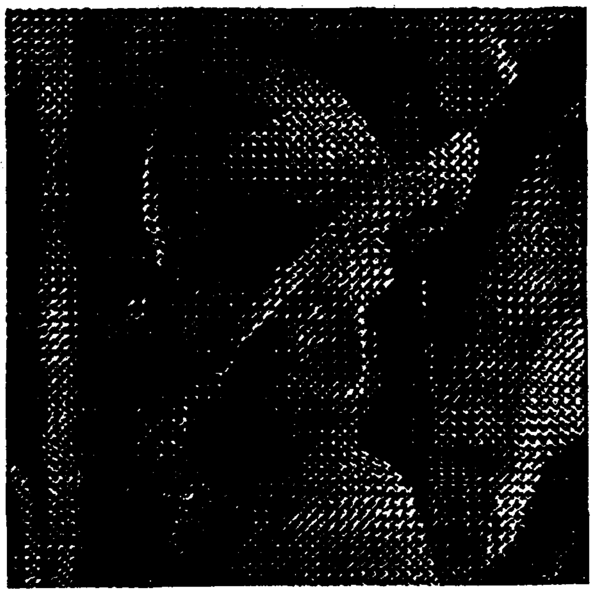


(a) Original image. (b) Degraded image. (c) Results after 6 iterations.

Figure 1: Restoration of noisy blurred synthetic image.

(a) Original girl image.


(b) Image degraded by 5 × 5 uniform blur and quantization noise


(c) Rsetored image using inverse filter.

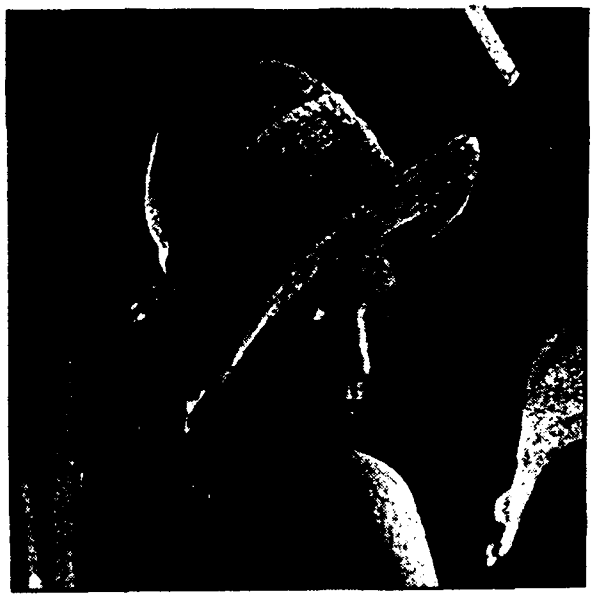

(d) Restored image using our approach.

Figure 2: Restoration of noisy blurred real image and comparison.

Potential Difference Learning and Its Optical Architecture

C. H. Wang and B. K. Jenkins

Sig nal and Image Processing Institute, Department of Electrical Engineering,
University of Southern California, Los Angeles, CA 90089-0272

ABSTRACT

A learning algorithm based on temporal difference of membrane potential of the neuron is proposed for self-organizing neural networks. It is independent of the neuron nonlinearity, so it can be applied to analog or binary neurons. Two simulations for learning of weights are presented; a single layer fully-connected network and a 3-layer network with hidden units for a distributed semantic network. The results demonstrate that this potential difference learning (PDL) can be used with neural architectures for various applications. Unlearning based on PDL for the single layer network is also discussed. Finally, an optical implementation of PDL is proposed.

1. INTRODUCTION

Most of the unsupervised learning algorithms are based on Hebb's hypothesis [1], which depends on the correlated activity of the pre- and postsynaptic nerve cells. For steady input patterns, Hebb's rule will suffer from weight overflow. Von der Malsburg (1973) [2] solved this by adding the constraint that the sum of the weights of a neuron is constant. This concept led to competitive learning, developed by Grossberg (1976) [3], and Rumelhart and Zipser (1985) [4]. They also assumed a winner-take-all algorithm (Fukushima 1975 [5]) to enhance the synaptic weight modification between neurons. Biologically, the sum of the weights of a neuron can likely be changed by the supply of some chemical substance. In this paper we propose a learning algorithm, potential difference learning (PDL), based on temporal difference of the neuron membrane potential. Because PDL is based on the membrane potential, it is independent of the nonlinear threshold function of the neuron. Its temporal characteristic prevents weight overflow and permits unlearning without access to individual weights.

In an artificial neural system, unlearning can provide for real time reprogramming and modification of the distributed storage for stable recollection, or equivalently, modification of the energy surface in an energy minimization problem. Hopfield proposed unlearning to reduce the accessibility of spurious states [6]. Our unlearning emphasizes reprogrammability and local modification of the energy surface for stable partial retrieval. The unlearning in PDL is done by presenting a sequence of patterns and global gain control; reversing the sign of the learning gain is not necessary. The distinction of learning and unlearning in PDL is in the data sequence and value of the gain constant for different phases.

The main advantages of potential difference learning are spontaneous learning without weight overflow for steady state input patterns and unlearning. Other features of PDL include contrast learning, temporally correlated and uncorrelated learning, learning independently of neuron type and ease of physical implementation.

2. POTENTIAL DIFFERENCE LEARNING AND ITS PROPERTIES

Like most learning rules, potential difference learning requires only local information for synapse modification. Given a neuron with n inputs, PDL is given by:

$$\underline{w}(k+1) = \Phi[\underline{w}(k) + K_a \alpha^{-1}(k) \cdot \Delta p(k) \cdot \underline{x}(k)] \quad (1)$$

$$\Delta p(k) \equiv \underline{w}^T(k)\underline{x}(k) - \underline{w}^T(k-1)\underline{x}(k-1) \quad (2)$$

$$y(k) = \Psi[\underline{w}^T(k)\underline{x}(k) - \theta(k)] \quad (3)$$

• Presented at SPIE's O-E Lase '88, Los Angeles, California, 10-15 January, 1988. "Neural Network Models for Optical Computing", SPIE vol. 882.

Where $\underline{w}(k)$ and $\underline{x}(k)$ are the input weights and stimuli respectively; they are represented as n x 1 vectors. $p(k)$ and $y(k)$ are the neuron potential and output value at time instant k. $\theta(k)$ is the threshold of the neuron and $K_a\alpha^{-1}(k)$ denotes the learning gain constant with K_a as the global gain constant and $\alpha^{-1}(k)$ as the adaptive gain constant. The weight $\underline{w}(k)$ is bounded by the function $\Phi(\cdot)$, which represents the physical limitation of synapses. Distinct from other learning models, PDL is independent of the output nonlinear function $\Psi(\cdot)$ of the neuron.

PDL has the following properties:

- (1). Self-organization: similar to Hebb's rule, PDL can modify the weights of synapses according to the input patterns.
- (2). Contrast learning: Weight modification is initiated by potential *difference*, which is caused by difference in input. Since most sensory preprocessing is differential in nature, PDL can provide a good approach for feature extraction when it is combined with neural architectures.
- (3). Unlearning: This can be used to erase stored states, to alter the energy surface or reprogram a network. PDL can provide unlearning capability by applying suitable pattern sequences to generate a negative potential difference Δp. This is discussed below.
- (4). Temporally correlated or uncorrelated learning: By varying the training sequences, the neuron can learn absolute patterns or temporal differences between training patterns. The temporal difference property may be useful in sensory information processing.
- (5). Ease of implementation: The PDL uses only local information to update the weights and only one differencer per neuron is needed to calculate the potential difference. The complexity is low when it is compared with differential Hebbian learning (Kosko 1986) [8] or drive-reinforcement learning (Klopf 1986) [9].
- (6). Independence from neuron nonlinearity: The learning rule is evoked by the potential change only, so various non-linear functions can be imposed on the neuron to make different types of neurons. Due to this feature, weight modification can still occur when the output is saturated or clamped as long as the weight is not saturated.

A variant of PDL is given by

$$\underline{w}(k+1) = \Phi[\underline{w}(k) + K_a\alpha^{-1}(k) \cdot \Delta y(k) \cdot \underline{x}(k)] \tag{4}$$

which replaces potential difference $\Delta p(k)$ with output difference $\Delta y(k)$. This can be used when the neuron potential is not physically available. The tradeoff is that weight modification no longer occurs when the neuron output is saturated. Equation (4) is similar in appearance to supervised learning (Widrow-Hoff rule), but here $\Delta y(k)$ refers to the temporal difference of the neuron output, instead of the spatial output error.

Other learning algorithms have been proposed based on the following:

$$\underline{w}(k+1) = \Phi[\underline{w}(k) + K_a\alpha^{-1}(k) \cdot \Delta y(k) \cdot \Delta\underline{x}(k)] \tag{5}$$

with Δ representing different forms of temporal difference [7], [8], [9]. The use of $\Delta\underline{x}(k)$ instead of $\underline{x}(k)$ and more complex definitions of time average in these learning rules causes a higher implementation complexity.

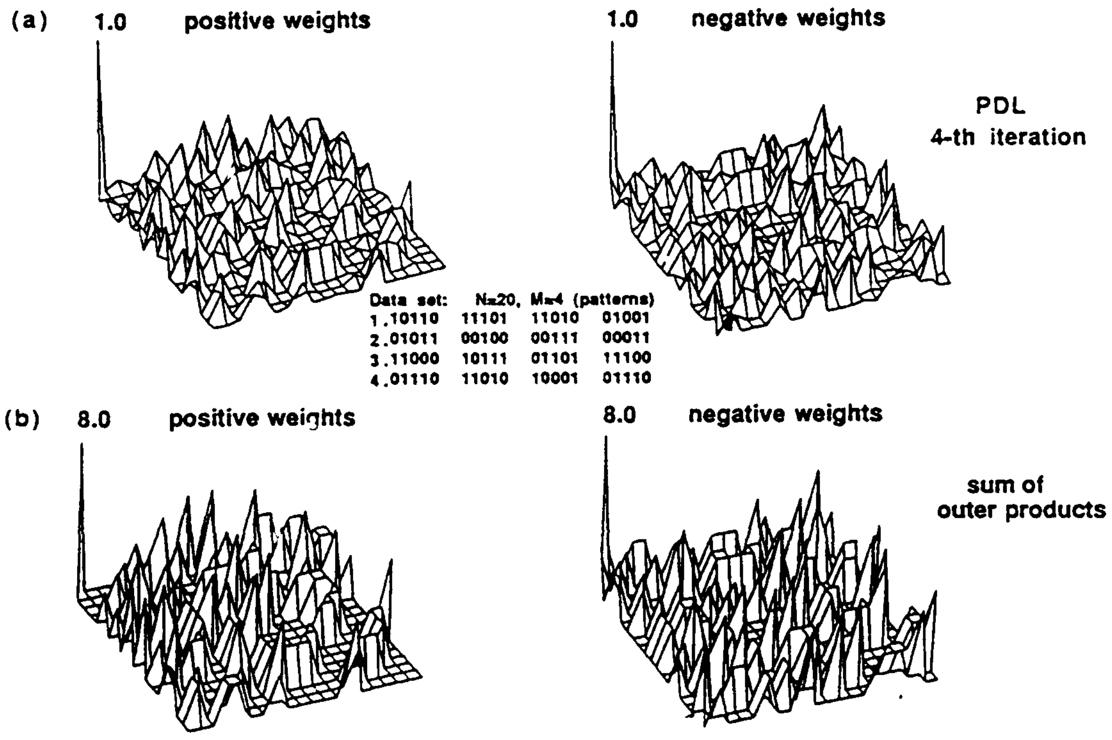
Due to the fact that the PDL rule is embedded in the neurons, we need some lateral interconnections between the neurons of the same layer to enhance the competitive or cooperative modification of synapses. One example is to use the winner-take-all algorithm [5]:

$$\underline{w}_j(k+1) = \Phi[\underline{w}_j(k) + K_a\alpha^{-1}(k) \cdot \Delta p_j(k) \cdot \underline{x}_j(k) \cdot \delta(y_j(k))] \tag{6}$$

where the subscript j denotes neuron j, and $\delta[y_j(k)] = 1$ if j^{th} neuron wins in his neighborhood, otherwise it is zero.

3. COMPUTER SIMULATIONS

First a single layer fully interconnected neural network, as described by Amari [10], Hopfield [11] and others, is simulated. Four input patterns [12], each 20 bits long, are presented to the external input of the network. The learning rule is our PDL with K_a=0.01 and $\alpha^{-1}(k)$=1. The neurons are binary with bipolar coding (+1,-1). The weights are initially set to zero. For each iteration, the four inputs were presented in sequence. Four iterations were performed. The resulting weight matrix is shown in Fig. 1 (a). PDL produces a near symmetric weight matrix, which is quite similar to the result obtained using the familiar sum of outer products, as shown in Fig. 1 (b). If a partial input is applied to the trained network, we can get full retrieval after several iterations, dependent on the hamming distance from the partial input.



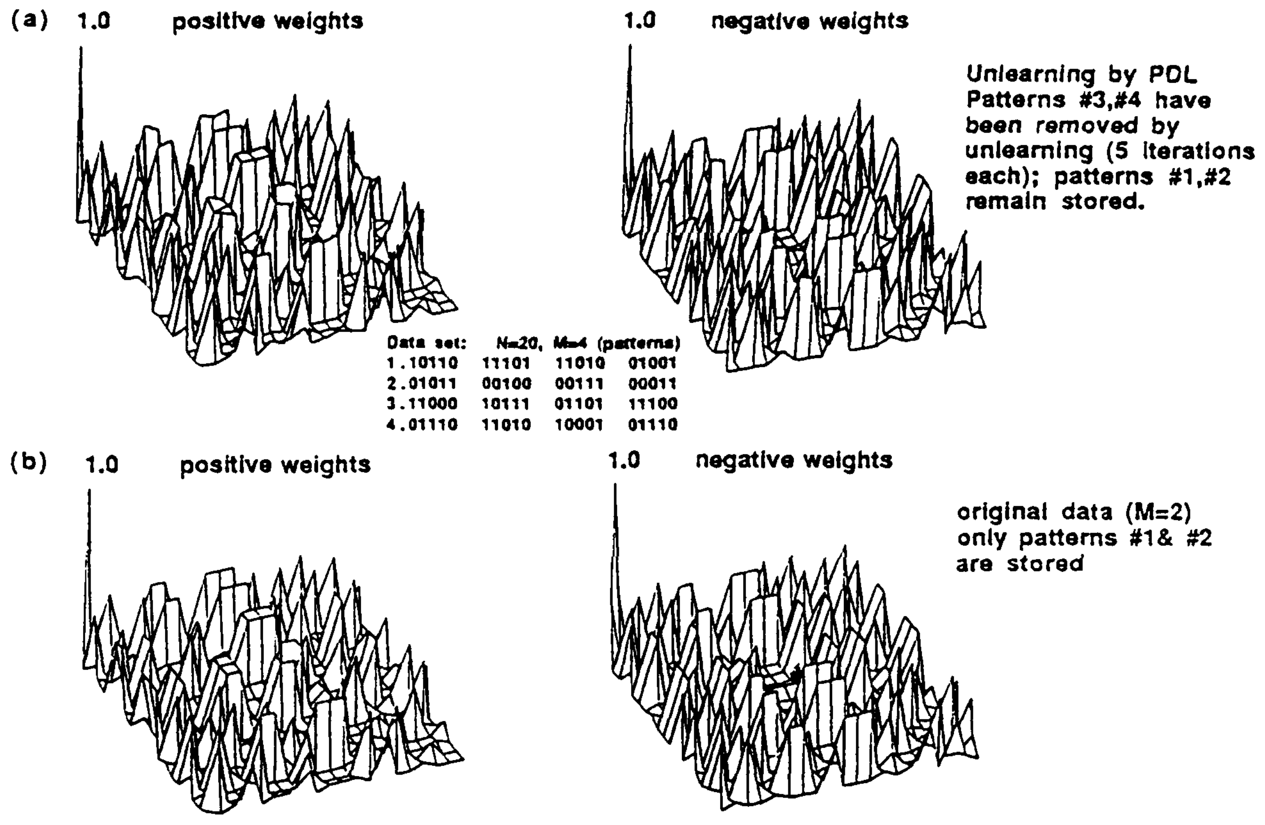
```
Data set:     N=20, M=4 (patterns)
1.10110  11101  11010  01001
2.01011  00100  00111  00011
3.11000  10111  01101  11100
4.01110  11010  10001  01110
```

Fig. 1 Comparison Between Outer Product T(i,j) Matrix and PDL
(20 neurons, 4 patterns)

The unlearning procedure of this network is divided into two stages. (1). Apply the data to be erased to the network with low (zero) global gain constant. (2). Use the same gain constant as for learning. In each step, present the input with one bit complemented; allow Δp to decrease to zero; restore that bit and complement the next bit for the next step. After all bits have been complemented, one iteration is completed. Starting with the trained weights of Fig. 1(a), two of the stored vectors (pattern #3 and #4) were erased using this unlearning procedure. We erase pattern #4 in five iterations, then erase pattern #3 in another five iterations. After each iteration, we test the convergence of the erased pattern. The resulting network would not converage to the erased states after just three iterations. For five iterations of unlearning, the weight matrix, Fig. 2(a), is very close to the original weight matrix that stored only pattern #1 and #2 as shown in Fig. 2(b). To measure the performance of unlearning, the resulting weight matrix is normalized by dividing it by a factor F, which is

$$F = \frac{\sqrt{\sum_{i,j} T_U^2(i,j)}}{\sqrt{\sum_{i,j} T_I^2(i,j)}} \tag{7}$$

where $T_U(i,j)$ is the resulting weight matrix after unlearning and $T_I(i,j)$ is the ideal weight matrix. Then



```
Data set:     N=20, M=4 (patterns)
1.10110   11101   11010   01001
2.01011   00100   00111   00011
3.11000   10111   01101   11100
4.01110   11010   10001   01110
```

Fig. 2 Unlearning of PDL from M=4 to M=2, N=20. Weight matrices (a) after unlearning, (b) ideal result.

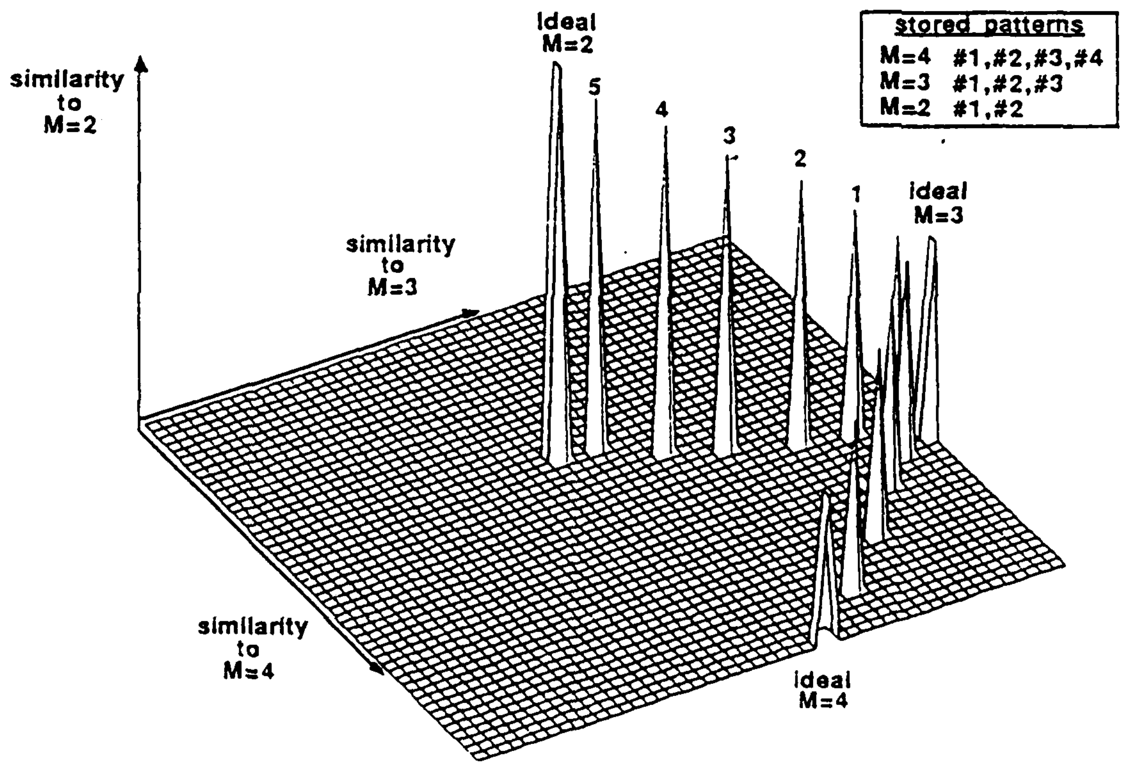


Fig. 3 Unlearning of PDL from M=4 to M=2 , N=20. Plot of similarity measures after each iteration. (5 iterations total for each unlearned pattern; numbered in sequence for unlearning of pattern #3). The ideal expected results for M=3 and M=2 are labelled.

a similarity measure, which is defined as the ratio of the matrix 1-norm [13] of these two weight matrices, is applied to evaluate the performance. Fig. 3 shows the similarity measure of the weight matrix after each iteration when unlearning using PDL from initially M=4 stored vectors to M=2 stored vectors.

The second example is a 3-layer network with two fully interconnected visible layers and a hidden layer. The purpose of this network is to do associative mapping between two visible layers by using one hidden layer. The visible layers are fully connected and the interconnections between the visible layers and the hidden layer are shown in Fig. 4. The visible layers use binary neurons to interface with the environment, while the hidden layer uses analog neurons. One neuron is used to calculate the average output, $u(k)$, of the hidden layer; then the competitive network of Fig.5, which is used in the hidden layer, reinforces those neurons with stronger output.

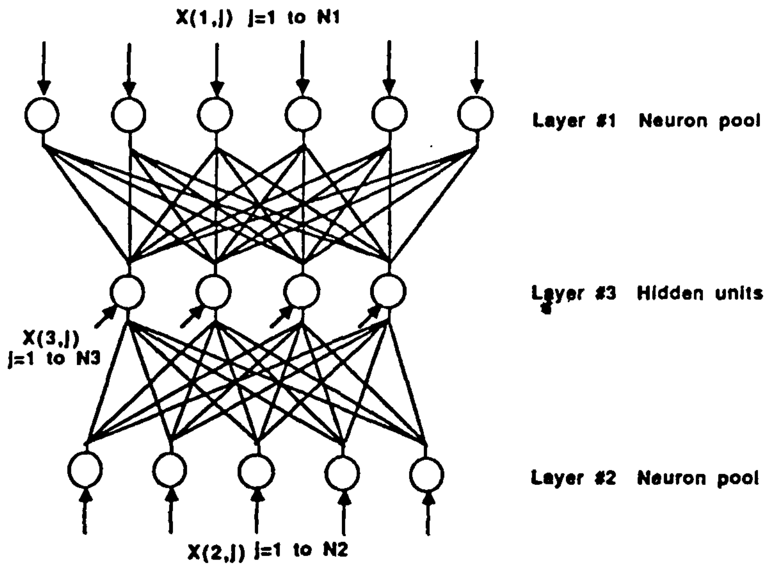


Fig. 4 Interconnections between hidden layer and visible layers. This interconnections are bidirectional with possibly different weights in each direction. Each visible layer is fully connected.

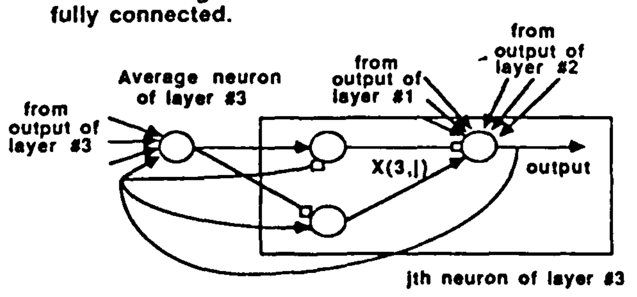


Fig. 5 competitive interconnections of the hidden layer.

The operation of the hidden layer is

$$u(k) = \sum_{j=1}^{N_3} \frac{1}{N_3} y_j^{(3)}(k) \tag{8}$$

$$y_j^{(3)}(k+1) = \psi\{\sum_{i=1}^{N_1} w_{ij}^{(1)} y_i^{(1)}(k) + \sum_{i=1}^{N_2} w_{ij}^{(2)} y_i^{(2)}(k) + \psi[y_j^{(3)}(k) - u(k)] - \psi[u(k) - y_j^{(3)}(k)]\} \tag{9}$$

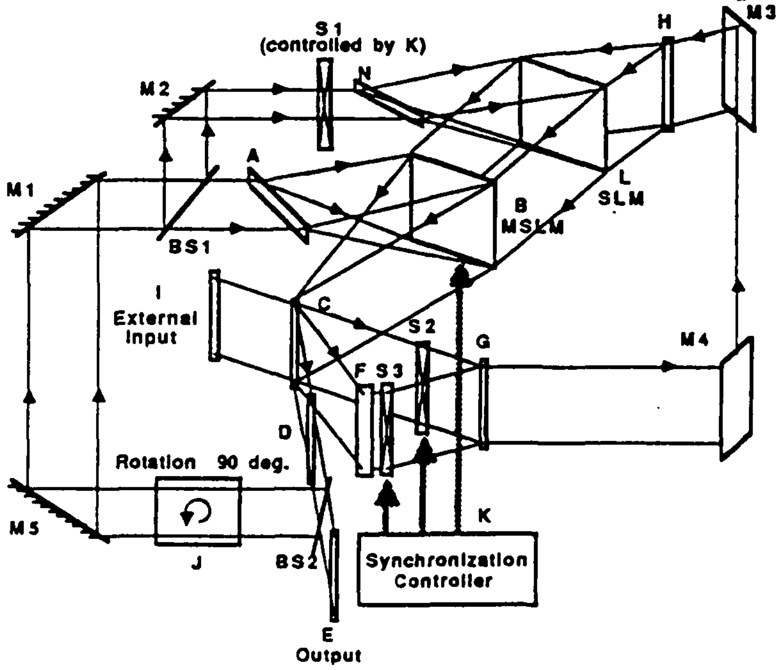
where $\psi(x)$ is 1 for $x \geq 1$, and is 0 for $x < 0$, else it is x. Superscripts denote the layer number and N_1, N_2, N_3 represent the number of neurons in layer 1, 2, and 3. $w_{ij}^{(l)}$ for $l = 1, 2$ represents the weight from the i^{th} neuron of layer l to the j^{th} neuron of the hidden layer. Initially, the weights of the visible layers are set to zero and the weights between the visible layers and the hidden layer are set to small random values. The visible layers are trained separately during the first phase, while the learning gain of the hidden layer is set to zero. In the second phase, the learning gain of the hidden layer and the visible layers is nonzero; we apply the corresponding patterns at these visible layers to train the hidden layer and the visible layers. After the learning phases, applying a partial input at the visible layer #1 will retrieve the full information at the same layer and associated data at the other visible layer. We have performed computer simulation of this network with 20 neurons in layer #1, 16 neurons in layer #2, and 10 neurons in the hidden layer. Eight patterns were stored, four into layer #1 and four into layer #2, and associations between pairs of these patterns were learned. The network randomly selected a set of one or more "representation" neurons in the hidden layer to form an association between each pair of patterns. Some of the sets of neurons for different pairs of patterns were disjoint, and some were partially overlapping. Table 1 shows the hidden neurons selected by the network for each associated pair of patterns. The last column in the table shows the pattern retrieved upon presentation of each layer #1 pattern. Since each set of representation neurons usually consists of multiple neurons, some fault tolerance is provided. However, when these sets overlap some interference can result during retrieval of associated patterns. This imperfect mapping results from the "soft" competitive network that was used.

Table 1 Simulation results of network of Fig. 4 and Fig. 5.

pattern stored in layer #1	resulting representation neurons in hidden layer	pattern stored in layer #2	pattern retrieved in layer #2
$\underline{a}_1$	1,4	$\underline{b}_1$	$\underline{b}_1$
$\underline{a}_2$	2,9	$\underline{b}_2$	$\underline{b}_2$
$\underline{a}_3$	2,3,5	$\underline{b}_3$	$\underline{b}_3$
$\underline{a}_4$	1,4,6	$\underline{b}_4$	$\underline{b}_1$

4. OPTICAL ARCHITECTURE OF PDL

A conceptual diagram of an optical implementation of PDL is shown in Fig. 6; it is somewhat similar to Fisher's associative processor [14]. Two spatial light modulators (SLMs) are used, one for storage of the weight matrix and one for generation of $\Delta\underline{w}(k)$. In addition, two 1-D storage devices and one 1-D threshold device are used.



S1-3	shutter
M1-5	mirrors
BS1-2	beam splitters
A	iteration input
B	microchannel SLM
C	potential output
D	1-D threshold array
E	output
F	optical delay line or storage
G	beam combiner
H	potential output $\underline{p}(k)$ or $\underline{p}(k-1)$
I	external data input
J	rotation optics
K	synchronization controller
L	SLM
N	1-D storage SLM for $\underline{x}(k)$

Fig. 6 Conceptual diagram of an optical implementation of potential difference learning.

"A" is the input $\underline{x}(k)$ from the previous iteration, which is expanded vertically to illuminate the weight storage "B". The reflected output from the microchannel SLM "B" is collected horizontally. This represents a vector, each component of which is $\sum w_{ij}x_j$. It is then combined with external input "I" to produce potential output at position "C". The output at "C" is split into three. The first one passes through 1-D threshold device "D" to generate outputs of the neurons. The second output of "C" passes through delay element F and shutter S3, yielding $\underline{p}(k-1)$ at "G". The third output path from "C" passes through shutter S2 to yield $\underline{p}(k)$ at "G". Only one of the shutter arrays S2 and S3 can be turned on at a time. "G" is a beam combiner and its output, either $\underline{p}(k)$ or $\underline{p}(k-1)$, reflects off mirrors M4, M3 and is expanded horizontally to illuminate the write side of SLM "L". A beam with intensity $\underline{x}(k)$ illuminates the read side of SLM "L" (which is read in reflection), to form outerproduct $\underline{p}(k)\underline{x}^T(k)$ or $\underline{p}(k-1)\underline{x}^T(k)$ for a 1-D array of neurons. At the first phase, $\underline{p}(k)\underline{x}^T(k)$ is added to the storage SLM "B". Then $\underline{p}(k-1)\underline{x}^T(k)$ is applied to "B", which is operated in subtraction mode during the second phase. These two steps calculate the potential difference and update the weights stored in "B".

During retrieval phase, partial input is applied to external input "I" and is then passed through threshold device "D", rotation optics "J", mirror M5, M1 and beam splitter BS1 to position "A" to perform vector-matrix computation of potential. Part of the iterated feedback signal $\underline{x}(k)$ reflects off BS1, M2 and is enabled by shutter S1 to store in 1-D storage SLM "N", which is used to form the outerproduct during the learning phase. Mirrors M1 and M5 are used, as shown, to implement feedback within a single layer network. For a multilayer network M1 and M5 can be removed (or replaced with beamsplitters) to send outputs to and receive signals from other layers.

5. CONCLUSIONS

This PDL provides a number of interesting features along with a moderate implementation complexity. It is a general technique that can be applied to different neuron types and different network models. Our simulations indicate that it learns correctly in a variety of networks. We also described an unlearning technique for the case of a fully connected network used as an associative memory, which does not require any sign reversal of the learning gain or any global access to the weights. Applications of PDL include low level processing such as extraction of features.

References

[1] D. O. Hebb, "Organization of behavior", *John Wiley & sons* press, (1949)

[2] Chr. von der Malsburg, "Self-organization of orientation sensitive cells in the striate cortex", *Kybernetik*, 14, 85-100 (1973)

[3] S. Grossberg, "Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors", *Biol. Cybernetics*, 23, 121-134 (1976)

[4] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning", *Cognitive Science*, 9, 75-112 (1985)

[5] K. Fukushima, "Cognitron: A self-organizing multilayered neural network", *Biol. Cybernetics*, 20, 121-136 (1975)

[6] J. J. Hopfield, D. I. Feinstein and R. G. Palmer, "Unlearning has a stabilizing effect in collective memories", *Nature*, 304, 158-159, July (1983)

[7] G. Chauvet, "Habituation rules for a theory of the cerebellar cortex", *Biol. Cybernetics*, 55, 201-209 (1986)

[8] Bart Kosko, "Differential Hebbian Learning", *Proc. of conf. on Neural Networks for Computering*, Snowbird, Utah, 277-282 (1986)

[9] A. H. Klopf, "A drive-reinforcement model of single neuron function: an alternative to the Hebbian neuronal model", *Proc. of conf. on Neural Networks for Computering*, Snowbird, Utah, 265-269 (1986)

[10] S-I. Amari, "Learning patterns and pattern sequences by self-organizing nets of threshold elements", *IEEE trans. on Computers*, C-21(11), 1197-1206 (1972)

[11] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational ability", *Proc. Natl. Acad. Sci. USA*, 79, 2554-2558 (1982)

[12] N. H. Farhat, D. Psaltis, A. Prata and E. Paek, "Optical implementation of the Hopfield model", *Appl. Optics*, 24, 1469-1475 (1985)

[13] C. T. Chen, Linear System Theory and Design, 57-59, Holt, Rinehart and Winston, 1984

[14] A. D. Fisher et. al., "Implementation of adaptive associative optical computing elements", *Proc. SPIE*, 625, 196-204 (1986)

T-ASSP/36/7//21366

Image Restoration Using a Neural Network

Yi-Tong Zhou
Rama Chellappa
Aseem Vaid
B. Keith Jenkins

Reprinted from
IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING
Vol. 36, No. 7, July 1988

Image Restoration Using a Neural Network

YI-TONG ZHOU, STUDENT MEMBER, IEEE, RAMA CHELLAPPA, SENIOR MEMBER, IEEE, ASEEM VAID, AND B. KEITH JENKINS, MEMBER, IEEE

Abstract—**A new approach for restoration of gray level images degraded by a known shift-invariant blur function and additive noise is presented using a neural computational network. A neural network model is employed to represent a possibly nonstationary image whose gray level function is the simple sum of the neuron state variables. The restoration procedure consists of two stages: estimation of the parameters of the neural network model and reconstruction of images. During the first stage, the parameters are estimated by comparing the energy function of the network to a constrained error function. The nonlinear restoration method is then carried out iteratively in the second stage by using a dynamic algorithm to minimize the energy function of the network. Owing to the model's fault-tolerant nature and computation capability, a high-quality image is obtained using this approach. A practical algorithm with reduced computational complexity is also presented. Several computer simulation examples involving synthetic and real images are given to illustrate the usefulness of our method. The choice of the boundary values to reduce the ringing effect is discussed, and comparisons to other restoration methods such as the SVD pseudoinverse filter, minimum mean-square error (MMSE) filter, and modified MMSE filter using the Gaussian Markov random field model are given. Finally, a procedure for learning the blur parameters from prototypes of original and degraded images is outlined.**

I. Introduction

RESTORATION of a high-quality image from a degraded recording is an important problem in early vision processing. Restoration techniques are applied to remove 1) system degradations such as blur due to optical system aberrations, atmospheric turbulence, motion, and diffraction; and 2) statistical degradations due to noise. Over the last 20 years, various methods such as the inverse filter [1], Wiener filter [1], Kalman filter [2], SVD pseudoinverse [1], [3], and many other model-based approaches have been proposed for image restorations. One of the major drawbacks of most of the image restoration algorithms is the computational complexity, so much so that many simplifying assumptions such as wide sense stationarity (WSS), availability of second-order image statistics have been made to obtain computationally feasible algorithms. The inverse filter method works only for extremely high signal-to-noise ratio images. The Wiener filter is usually implemented only after the wide sense stationary assumption has been made for images. Furthermore, knowledge of the power spectrum or correlation matrix of the undegraded image is required. Often times, additional assumptions regarding boundary conditions are made so that fast orthogonal transforms can be used. The Kalman filter approach can be applied to nonstationary image, but is computationally very intensive. Similar statements can be made for the SVD pseudoinverse filter method. Approaches based on noncausal models such as the noncausal autoregressive or Gauss Markov random field models [4], [5] also make assumptions such as WSS and periodic boundary conditions. It is desirable to develop a restoration algorithm that does not make WSS assumptions and can be implemented in a reasonable time. An artificial neural network system that can perform extremely rapid computations seems to be very attractive for image restoration in particular and image processing and pattern recognition [6] in general.

In this paper, we use a neural network model containing redundant neurons to restore gray level images degraded by a known shift-invariant blur function and noise. It is based on the method described in [7]–[9] using a simple sum number representation [10]. The image gray levels are represented by the simple sum of the neuron state variables which take binary values of 1 or 0. The observed image is degraded by a shift-invariant function and noise. The restoration procedure consists of two stages: estimation of the parameters of the neural network model and reconstruction of images. During the first stage, the parameters are estimated by comparing the energy function of the neural network to the constrained error function. The nonlinear restoration algorithm is then implemented using a dynamic iterative algorithm to minimize the energy function of the neural network. Owing to the model's fault-tolerant nature and computation capability, a high-quality image is obtained using this approach. In order to reduce computational complexity, a practical algorithm, which has equivalent results to the original one suggested above, is developed under the assumption that the neurons are sequentially visited. We illustrate the usefulness of this approach by using both synthetic and real images degraded by a known shift-invariant blur function with or without noise. We also discuss the problem of choosing boundary values and introduce two methods to reduce the ringing effect. Comparisons to other restoration methods such as the SVD pseudoinverse filter, the minimum mean-square error (MMSE) filter, and the modified MMSE filter using a Gaussian Markov random field model are given using real images. The advantages of the method developed in this paper are: 1) WSS assumption is not required

Manuscript received February 22, 1988. This work was supported in part by AFOSR Contract F-49620-87-C-0007 and AFOSR Grant 86-0196.

The authors are with the Signal and Image Processing Institute, Department of Electrical Engineering—Systems, University of Southern California, Los Angeles, CA 90089.

IEEE Log Number 8821366.

for the images, 2) it can be implemented rapidly, and 3) it is fault tolerant.

In the above, the interconnection strengths (also called weights) of the neural network for image restoration are known from the parameters of the image degradation model and the smoothing constraints. We also consider learning of the parameters for the image degradation model and formulate it as a problem of computing the parameters from samples of the original and degraded images. This is implemented as a secondary neural network. A different scheme is used to represent multilevel activities for the parameters; some of its properties are complementary to those of the simple sum scheme. The learning procedure is accomplished by running a greedy algorithm. Some results of learning the blur parameters are presented using synthetic and real image examples.

The organization of this paper is as follows. A network model containing redundant neurons for image representation and the image degradation model is given in Section II. A technique for parameter estimation is presented in Section III. Image generation using a dynamic algorithm is described in Section IV. A practical algorithm with reduced computational complexity is presented in Section V. Computer simulation results using synthetic and real degraded images are given in Section VI. Choice of the boundary values is discussed in Section VII. Comparisons to other methods are given in Section VIII. A procedure for learning the blur parameters from prototypes of original and degraded images is outlined in Section IX, and conclusions and remarks are included in Section X.

II. A Neural Network for Image Representation

We use a neural network containing redundant neurons for representing the image gray levels. The model consists of $L^2 \times M$ mutually interconnected neurons where L is the size of image and M is the maximum value of the gray level function. Let $V = \{v_{i,k}$ where $1 \le i \le L^2$, $1 \le k \le M\}$ be a binary state set of the neural network with $v_{i,k}$ (1 for firing and 0 for resting) denoting the state of the (i, k)th neuron. Let $T_{i,k;j,l}$ denote the strength (possibly negative) of the interconnection between neuron (i, k) and neuron (j, l). We require symmetry:

$$T_{i,k;j,l} = T_{j,l;i,k} \quad \text{for } 1 \le i, j \le L^2 \text{ and}$$
$$1 \le l, k \le M.$$

We also allow for neurons to have self-feedback, i.e., $T_{i,k;i,k} \neq 0$. In this model, each neuron (i, k) randomly and asynchronously receives inputs $\Sigma T_{i,k;j,l} v_{j,l}$ from all neurons and a bias input $I_{i,k}$:

$$u_{i,k} = \sum_{j}^{L^2} \sum_{l}^{M} T_{i,k;j,l} v_{j,l} + I_{i,k}. \tag{1}$$

Each $u_{i,k}$ is fed back to corresponding neurons after thresholding:

$$v_{i,k} = g(u_{i,k}) \tag{2}$$

where $g(x)$ is a nonlinear function whose form can be taken as

$$g(x) = \begin{cases} 1 & \text{if } x \ge 0 \\ 0 & \text{if } x < 0. \end{cases} \tag{3}$$

In this model, the state of each neuron is updated by using the latest information about other neurons.

The image is described by a finite set of gray level functions $\{x(i, j)$ where $1 \le i, j \le L\}$ with $x(i, j)$ (positive integer number) denoting the gray level of the pixel (i, j). The image gray level function can be represented by a simple sum of the neuron state variables as

$$x(i, j) = \sum_{k=1}^{M} v_{m,k} \tag{4}$$

where $m = (i - 1) \times L + j$. Here the gray level functions have degenerate representations. Use of this redundant number representation scheme yields advantages such as fault tolerance and faster convergence to the solution [10].

By using the lexicographic notation, the image degradation model can be written as

$$Y = HX + N \tag{5}$$

where H is the "blur matrix" corresponding to a blur function, N is the signal independent white noise, and X and Y are the original and degraded images, respectively. Furthermore, H and N can be represented as

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,L^2} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,L^2} \\ \vdots & \vdots & \cdots & \vdots \\ h_{L^2,1} & h_{L^2,2} & \cdots & h_{L^2,L^2} \end{bmatrix} \tag{6}$$

and

$$N = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_L \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_{L^2} \end{bmatrix},$$

$$N_i = \begin{bmatrix} n(i, 1) \\ n(i, 2) \\ \vdots \\ n(i, L) \end{bmatrix} = \begin{bmatrix} n_{(i-1)\times L+1} \\ n_{(i-1)\times L+2} \\ \vdots \\ n_{i\times L} \end{bmatrix} \tag{7}$$

respectively. Vectors X and Y have similar representations. Equation (5) is similar to the simultaneous equations solution of [10], but differs in that it includes a noise term.

The shift-invariant blur function can be written as a convolution over a small window, for instance, it takes the form

$$h(k, l) = \begin{cases} \frac{1}{2} & \text{if } k = 0, l = 0 \\ \frac{1}{16} & \text{if } |k|, |l| \le 1, (k, l) \neq (0, 0); \end{cases} \tag{8}$$

accordingly, the "blur matrix" H will be a block Toeplitz or block circulant matrix (if the image has periodic boundaries). The block circulant matrix corresponding to (8) can be written as

$$H = \begin{bmatrix} H_0 & H_1 & \mathbf{0} & \cdots & \mathbf{0} & H_1 \\ H_1 & H_0 & H_1 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ H_1 & \mathbf{0} & \mathbf{0} & \cdots & H_1 & H_0 \end{bmatrix} \tag{9}$$

where

$$H_0 = \begin{bmatrix} \frac{1}{2} & \frac{1}{16} & 0 & \cdots & 0 & \frac{1}{16} \\ \frac{1}{16} & \frac{1}{2} & \frac{1}{16} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \frac{1}{16} & 0 & 0 & \cdots & \frac{1}{16} & \frac{1}{2} \end{bmatrix},$$

$$H_1 = \begin{bmatrix} \frac{1}{16} & \frac{1}{16} & 0 & \cdots & 0 & \frac{1}{16} \\ \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \frac{1}{16} & 0 & 0 & \cdots & \frac{1}{16} & \frac{1}{16} \end{bmatrix} \tag{10}$$

and $\mathbf{0}$ is null matrix whose elements are all zeros.

III. Estimation of Model Parameters

The neural model parameters, the interconnection strengths, and bias inputs can be determined in terms of the energy function of the neural network. As defined in [7], the energy function of the neural network can be written as

$$E = -\frac{1}{2} \sum_{i=1}^{L^2} \sum_{j=1}^{L^2} \sum_{k=1}^{M} \sum_{l=1}^{M} T_{i,k;j,l} v_{i,k} v_{j,l} - \sum_{i=1}^{L^2} \sum_{k=1}^{M} I_{i,k} v_{i,k}. \tag{11}$$

In order to use the spontaneous energy-minimization process of the neural network, we reformulate the restoration problem as one of minimizing an error function with constraints defined as

$$E = \frac{1}{2} \| Y - H\hat{X} \|^2 + \frac{1}{2} \lambda \| D\hat{X} \|^2 \tag{12}$$

where $\|Z\|$ is the L_2 norm of Z and λ is a constant. Such a constrained error function is widely used in the image restoration problems [1] and is also similar to the regularization techniques used in early vision problems [11]. The first term in (12) is to seek an $\hat{X}$ such that $H\hat{X}$ approximates Y in a least squares sense. Meanwhile, the second term is a smoothness constraint on the solution $\hat{X}$. The constant λ determines their relative importance to achieve both noise suppression and ringing reduction.

In general, if H is a low-pass distortion, then D is a high-pass filter. A common choice of D is a second-order differential operator which can be approximated as a local window operator in the 2-D discrete case. For instance, if D is a Laplacian operator

$$\nabla = \frac{\partial^2}{\partial i^2} + \frac{\partial^2}{\partial j^2} \tag{13}$$

it can be approximated as a window operator

$$\frac{1}{6} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}. \tag{14}$$

Then D will be a block Toeplitz matrix similar to (9).

Expanding (12) and then replacing x_i by (4), we have

$$\begin{aligned} E &= \frac{1}{2} \sum_{p=1}^{L^2} \left(y_p - \sum_{i=1}^{L^2} h_{p,i} x_i \right)^2 + \frac{1}{2} \lambda \sum_{p=1}^{L^2} \left(\sum_{i=1}^{L^2} d_{p,i} x_i \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^{L^2} \sum_{j=1}^{L^2} \sum_{k=1}^{M} \sum_{l=1}^{M} \sum_{p=1}^{L^2} h_{p,i} h_{p,j} v_{i,k} v_{j,l} \\ &\quad + \frac{1}{2} \lambda \sum_{i=1}^{L^2} \sum_{j=1}^{L^2} \sum_{k=1}^{M} \sum_{l=1}^{M} \sum_{p=1}^{L^2} d_{p,i} d_{p,j} v_{i,k} v_{j,l} \\ &\quad - \sum_{i=1}^{L^2} \sum_{k=1}^{M} \sum_{p=1}^{L^2} y_p h_{p,i} v_{i,k} + \frac{1}{2} \sum_{p=1}^{L^2} y_p^2. \end{aligned} \tag{15}$$

By comparing the terms in (15) to the corresponding terms in (11) and ignoring the constant term $\frac{1}{2}\sum_{p=1}^{L^2} y_p^2$, we can determine the interconnection strengths and bias inputs as

$$T_{i,k;j,l} = -\sum_{p=1}^{L^2} h_{p,i} h_{p,j} - \lambda \sum_{p=1}^{L^2} d_{p,i} d_{p,j} \tag{16}$$

and

$$I_{i,k} = \sum_{p=1}^{L^2} y_p h_{p,i} \tag{17}$$

where $h_{i,j}$ and $d_{i,j}$ are the elements of the matrices H and D, respectively. Two interesting aspects of (16) and (17) should be pointed out: 1) the interconnection strengths are independent of subscripts k and l and the bias inputs are independent of subscript k, and 2) the self-connection $T_{i,k;i,k}$ is not equal to zero which requires self-feedback for neurons.

From (16), one can see that the interconnection strengths are determined by the shift-invariant blur function, differential operator, and constant λ. Hence, $T_{i,k;j,l}$ can be computed without error provided the blur function is known. However, the bias inputs are functions of the observed degraded image. If the image is degraded by a shift-invariant blur function only, then $I_{i,k}$ can be estimated perfectly. Otherwise, $I_{i,k}$ is affected by noise. The reasoning behind this statement is as follows. By replacing y_p by $\sum_{i=1}^{L^2} h_{p,i}x_i + n_p$, we have

$$I_{i,k} = \sum_{p=1}^{L^2} \left(\sum_{i=1}^{L^2} h_{p,i}x_i + n_p \right) h_{p,i}$$
$$= \sum_{p=1}^{L^2} \sum_{i=1}^{L^2} h_{p,i}x_i h_p + \sum_{p=1}^{L^2} n_p h_{p,i}. \tag{18}$$

The second term in (18) represents the effects of noise. If the signal-to-noise ratio (SNR), defined by

$$\text{SNR} = 10 \log_{10} \frac{\sigma_s^2}{\sigma_n^2} \tag{19}$$

where σ_s^2 and σ_n^2 are variances of signal and noise, respectively, is low, then we have to choose a large λ to suppress effects due to noise. It seems that in the absence of noise, the parameters can be estimated perfectly, ensuring exact recovery of the image as error function E tends to zero. However, the problem is not so simple because the restoration performance depends on both the parameters and the blur function when a mean-square error or least square error such as (12) is used. A discussion about the effect of blur function is given in Section X.

IV. Restoration

Restoration is carried out by neuron evaluation and an image construction procedure. Once the parameters $T_{i,k;j,l}$ and $I_{i,k}$ are obtained using (16) and (17), each neuron can randomly and asynchronously evaluate its state and readjust accordingly using (1) and (2). When one quasi-minimum energy point is reached, the image can be constructed using (4).

However, this neural network has self-feedback, i.e., $T_{i,k;i,k} \neq 0$. As a result, the energy function E does not always decrease monotonically with a transition. This is explained below. Define the state change $\Delta v_{i,k}$ of neuron (i, k) and energy change ΔE as

$$\Delta v_{i,k} = v_{i,k}^{\text{new}} - v_{i,k}^{\text{old}} \quad \text{and} \quad \Delta E = E^{\text{new}} - E^{\text{old}}.$$

Consider the energy function

$$E = -\frac{1}{2} \sum_{i=1}^{L^2} \sum_{j=1}^{L^2} \sum_{k=1}^{M} \sum_{l=1}^{M} T_{i,k;j,l} v_{i,k} v_{j,l} - \sum_{i=1}^{L^2} \sum_{k=1}^{M} I_{i,k} v_{i,k}. \tag{20}$$

Then the change ΔE due to a change $\Delta v_{i,k}$ is given by

$$\Delta E = -\left(\sum_{j=1}^{L^2} \sum_{l=1}^{M} T_{i,k;j,l} v_{j,l} + I_{i,k} \right) \Delta v_{i,k} - \frac{1}{2} T_{i,k;i,k} (\Delta v_{i,k})^2 \tag{21}$$

which is not always negative. For instance, if

$$v_{i,k}^{\text{old}} = 0, \quad u_{i,k} = \sum_{j=1}^{L^2} \sum_{l=1}^{M} T_{i,k;j,l} v_{j,l} + I_{i,k} > 0$$

and the threshold function is as in (3), then $v_{i,k}^{\text{new}} = 1$ and $\Delta v_{i,k} > 0$. Thus, the first term in (21) is negative. But

$$T_{i,k;i,k} = -\sum_{p=1}^{L^2} h_{p,i}^2 - \lambda \sum_{p=1}^{L^2} d_{p,i}^2 < 0$$

with $\lambda > 0$, leading to

$$-\frac{1}{2} T_{i,k;i,k} (\Delta v_{i,k})^2 > 0.$$

When the first term is less than the second term in (21), then $\Delta E > 0$ (we have observed this in our experiment), which means E is not a Lyapunov function. Consequently, the convergence of the network is not guaranteed [12].

Thus, depending on whether convergence to a local minimum or a global minimum is desired, we can design a deterministic or stochastic decision rule. The deterministic rule is to take a new state $v_{i,k}^{\text{new}}$ of neuron (i, k) if the energy change ΔE due to state change $\Delta v_{i,k}$ is less than zero. If ΔE due to state change is > 0, no state change is affected. One can also design a stochastic rule similar to the one used in stimulated annealing techniques [13], [14]. The details of this stochastic scheme are given as follows.

Define a Boltzmann distribution by

$$\frac{p_{\text{new}}}{p_{\text{old}}} = e^{-\Delta E/T}$$

where p_{new} and p_{old} are the probabilities of the new and old global state, respectively, ΔE is the energy change, and T is the parameter which acts like temperature. A new state $v_{i,k}^{\text{new}}$ is taken if

$$\frac{p_{\text{new}}}{p_{\text{old}}} > 1 \text{ or if } \frac{p_{\text{new}}}{p_{\text{old}}} \leq 1 \text{ but } \frac{p_{\text{new}}}{p_{\text{old}}} > \xi$$

where ξ is a random number uniformly distributed in the interval [0, 1].

The restoration algorithm is summarized as below.

Algorithm 1:

1) Set the initial state of the neurons.

2) Update the state of all neurons randomly and asynchronously according to the decision rule.

3) Check the energy function; if energy does not change, go to step 4); otherwise, go back to step 2).

4) Construct an image using (4).

V. A Practical Algorithm

The algorithm described above is difficult to simulate on a conventional computer owing to high computational complexity, even for images of reasonable size. For instance, if we have an $L \times L$ image with M gray levels, then L^2M neurons and $\frac{1}{2}L^4M^2$ interconnections are required and L^4M^2 additions and multiplications are needed at each iteration. Therefore, the space and time complexities are $O(L^4M^2)$ and $O(L^4M^2K)$, respectively, where K, typically 10–100, is the number of iterations. Usually, L and M are 256–1024 and 256, respectively. However, simplification is possible if the neurons are sequentially updated.

In order to simplify the algorithm, we begin by reconsidering (1) and (2) of the neural network. As noted earlier, the interconnection strengths given in (16) are independent of subscripts k and l and the bias inputs given in (17) are independent of subscript k; the M neurons used to represent the same image gray level function have the same interconnection strengths and bias inputs. Hence, one set of interconnection strengths and one bias input are sufficient for every gray level function, i.e., the dimensions of the interconnection matrix T and bias input matrix I can be reduced by a factor of M^2. From (1), all inputs received by a neuron, say the (i, k)th neuron, can be written as

$$\begin{aligned} u_{i,k} &= \sum_{j}^{L^2} T_{i,\cdot;j,\cdot} \left(\sum_{l}^{M} v_{j,l} \right) + I_{i,\cdot} \\ &= \sum_{j}^{L^2} T_{i,\cdot;j,\cdot} x_j + I_{i,\cdot} \end{aligned} \tag{22}$$

where we have used (4) and x_j is the gray level function of the jth image pixel. The symbol "$\cdot$" in the subscripts means that the $T_{i,\cdot;j,\cdot}$ and $I_{i,\cdot}$ are independent of k. Equation (22) suggests that we can use a multivalue number to replace the simple sum number. Since the interconnection strengths are determined by the blur function, the differential operator, and the constant λ as shown in (16), it is easy to see that if the blur function is local, then most interconnection strengths are zeros and the neurons are locally connected. Therefore, most elements of the interconnection matrix T are zeros. If the blur function is shift invariant taking the form in (8), then the interconnection matrix is block Toeplitz so that only a few elements need to be stored. Based on the value of inputs $u_{i,k}$, the state of the (i, k)th neuron is updated by applying a decision rule. The state change of the (i, k)th neuron in turn causes the gray level function x_i to change:

$$x_i^{\text{new}} = \begin{cases} x_i^{\text{old}} & \text{if } \Delta v_{i,k} = 0 \\ x_i^{\text{old}} + 1 & \text{if } \Delta v_{i,k} = 1 \\ x_i^{\text{old}} - 1 & \text{if } \Delta v_{i,k} = -1 \end{cases} \tag{23}$$

where $\Delta v_{i,k} = v_{i,k}^{\text{new}} - v_{i,k}^{\text{old}}$ is the state change of the (i, k)th neuron. The superscripts "new" and "old" are for after and before updating, respectively. We use x_i to represent the gray level value as well as the output of M neurons representing x_i. Assuming that the neurons of the network are sequentially visited, it is straightforward to show that the updating procedure can be reformulated as

$$u_{i,k} = \sum_{j}^{L^2} T_{i,\cdot;j,\cdot} x_j + I_{i,\cdot} \tag{24}$$

$$\Delta v_{i,k} = g(u_{i,k}) = \begin{cases} \Delta v_{i,k} = 0 & \text{if } u_{i,k} = 0 \\ \Delta v_{i,k} = 1 & \text{if } u_{i,k} > 0 \\ \Delta v_{i,k} = -1 & \text{if } u_{i,k} < 0 \end{cases} \tag{25}$$

$$x_i^{\text{new}} = \begin{cases} x_i^{\text{old}} + \Delta v_{i,k} & \text{if } \Delta E < 0 \\ x_i^{\text{old}} & \text{if } \Delta E \geq 0. \end{cases} \tag{26}$$

Note that the stochastic decision rule can also be used in (26). In order to limit the gray level function to the range 0–255 after each updating step, we have to check the value of the gray level function x_i^{new}. Equations (24), (25), and (26) give a much simpler algorithm. This algorithm is summarized below.

Algorithm 2:

1) Take the degraded image as the initial value.

2) Sequentially visit all numbers (image pixels). For each number, use (24), (25), and (26) to update it repeatedly until there is no further change, i.e., if $\Delta v_{i,k} = 0$ or energy change $\Delta E \geq 0$; then move to the next one.

3) Check the energy function; if energy does not change anymore, a restored image is obtained; otherwise, go back to step 2) for another iteration.

The calculations of the inputs $u_{i,k}$ of the (i, k)th neuron and the energy change ΔE can be simplified furthermore. When we update the same image gray level function repeatedly, the input received by the current neuron (i, k) can be computed by making use of the previous result

$$u_{i,k} = u_{i,k-1} + \Delta v_{i,k} T_{i,\cdot;i,\cdot} \tag{27}$$

where $u_{i,k-1}$ is the inputs received by the $(i, k - 1)$th neuron. The energy change ΔE due to the state change of the (i, k)th neuron can be calculated as

$$\Delta E = -u_{i,k} \Delta v_{i,k} - \tfrac{1}{2} T_{i,\cdot;i,\cdot} (\Delta v_{i,k})^2. \tag{28}$$

If the blur function is shift invariant, all these simplifications reduce the space and time complexities significantly from $O(L^4M^2)$ and $O(L^4M^2K)$ to $O(L^2)$ and $O(ML^2K)$, respectively. Since every gray level function needs only a few updating steps after the first iteration, the computation at each iteration is $O(L^2)$. The resulting algorithm can be easily simulated on minicomputers for images as large as 512 × 512.

VI. Computer Simulations

The practical algorithm described in the previous section was applied to synthetic and real images on a Sun-3/160 Workstation. In all cases, only the deterministic decision rule was used. The results are summarized in Figs. 1 and 2.

Fig. 1 shows the results for a synthetic image. The original image shown in Fig. 1(a) is of size 32 × 32 with three gray levels. The image was degraded by convolving with a 3 × 3 blur function as in (8) using circulant boundary conditions; 22 dB white Gaussian noise was added after convolution. A perfect image was obtained after six

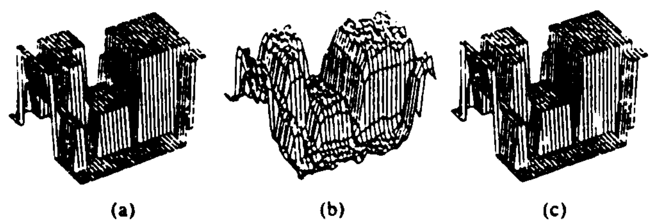


Fig. 1. Restoration of noisy blurred synthetic image. (a) Original image. (b) Degraded image. (c) Result after six iterations.

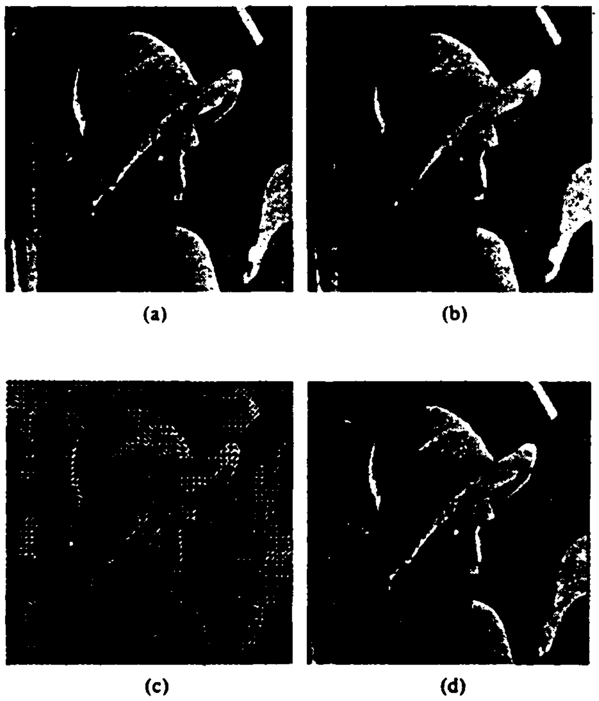


Fig. 2. Restoration of noisy blurred real image. (a) Original girl image. (b) Image degraded by 5 × 5 uniform blur and quantization noise. (c) The restored image using inverse filter. (d) The restored image using our approach.

iterations without preprocessing. We set the initial state of all neurons to equal 1, i.e., firing, and chose $\lambda = 0$ due to the well conditioning of the blur function.

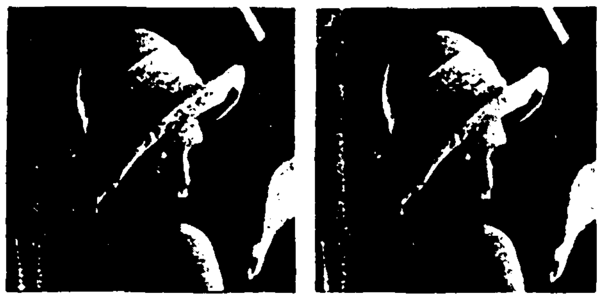
Fig. 2(a) shows the original girl image. The original image is of size 256 × 256 with 256 gray levels. The variance of the original image is 2797.141. It was degraded by a 5 × 5 uniform blur function. A small amount of quantization noise was introduced by quantizing the convolution results to 8 bits. The noisy blurred image is shown in Fig. 2(b). For comparison purpose, Fig. 2(c) shows the output of an inverse filter [15], completely overridden by the amplified noise and the ringing effects due to the ill-conditioned blur matrix H. Since the blur matrix H corresponding to the 5 × 5 uniform blur function is not singular, the pseudoinverse filter [15] and the inverse filter have the same output. The restored image by using our approach is shown in Fig. 2(d). In order to avoid the ringing effects due to the boundary conditions, we took 4 pixel wide boundaries, i.e., the first and last four rows and columns, from the original image and updated the interior region (248 × 248) of the image only. The noisy blurred image was used as an initial condition for accelerating the convergence. The constant λ was set to zero because of small noise and good boundary values. The restored image in Fig. 2(d) was obtained after 213 iterations. The square error (i.e., energy function) defined in (12) is 0.02543 and the square error between the original and the restored image is 66.5027.

VII. Choosing Boundary Values

As mentioned in [16], choosing boundary values is a common problem for techniques ranging from deterministic inverse filter algorithms to stochastic Kalman filters. In these algorithms, boundary values determine the entire solution when the blur is uniform [17]. The same problem occurs in the neural network approach. Since the 5 × 5 uniform blur function is ill conditioned, improper boundary values may cause ringing which may affect the restored image completely. For example, appending zeros to the image as boundary values introduces a sharp edge at the image border and triggers ringing in the restored image even if the image has zero mean. Another procedure is to assume a periodic boundary. When the left (top) and right (bottom) borders of the image are different, a sharp edge is formed and ringing results even though the degraded image has been formed by blurring with periodic boundary conditions. The drawbacks of these two assumptions for boundary values were reported in [16], [2], [18] for the 2-D Kalman filtering technique. We also tested our algorithm using these two assumptions for boundary values; the results indicate the restored images were seriously affected by ringing.

In the last section, to avoid the ringing effect, we took 4 pixel wide borders from the original image as boundary values for restoration. Since the original image is not available in practice always, an alternative to eliminate the ringing effect caused by sharp false edges is to use the blurred noisy boundaries from the degraded image. Fig. 3(a) shows the restored image using the first and last four rows and columns of the blurred noisy image in Fig. 2(b) as boundary values. In the restored image, there still exists some ringing due to the naturally occurring sharp edges in the region near the borders in the original image, but not due to boundary values. A typical cut of the restored image to illustrate ringing near the borders is shown in Fig. 4. To remove the ringing near the borders caused by naturally occurring sharp edges in the original image, we suggest the following techniques.

First, divide the image into three regions: border, subborder, and interior region as shown in Fig. 5. For the 5 × 5 uniform blur case, the border region will be 4 pixels wide due to the boundary effect of the bias input $I_{i,k}$ in (17), and the subborder region will be 4 or 8 pixels wide. In fact, the width of the subborder region will be image dependent. If the regions near the border are smooth, then the width of the subborder region will be small or even zero. If the border contains many sharp edges, the width will be large. For the real girl image, we chose the width



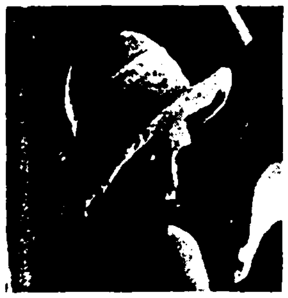


Fig. 3. Results using blurred noisy boundaries. (a) Blurred noisy boundaries. (b) Method 1. (c) Method 2.

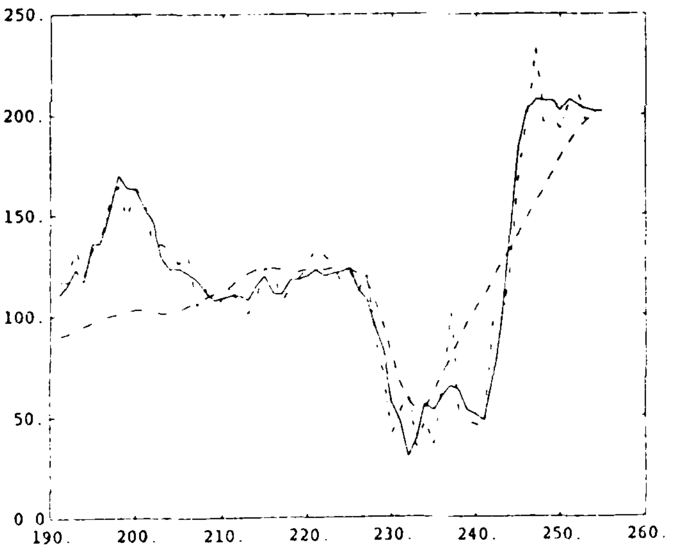


Fig. 4. One typical cut of the restored image using the blurred noisy boundaries. Solid line for original image, dashed line for blurred noisy image, and dashed and dotted line for restored image.

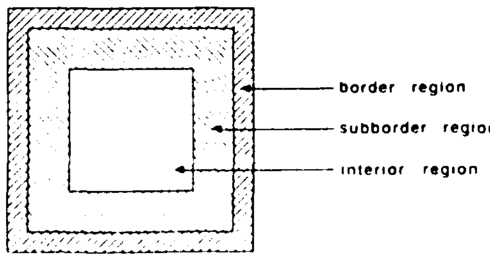


Fig. 5. Border, subborder, and interior regions of the image.

of the subborder region to be 8 pixels. We suggest using one of the following two methods.

Method 1: In the case of small noise, such as quantization error noise, the blurred image is usually smooth. Therefore, we restricted the difference between the restored and blurred image in the subborder region to a certain range to reduce the ringing effect. Mathematically, this constraint can be written as

$$\|\hat{x}_i - y_i\| \le T \qquad \text{for } i \in \text{subborder region} \quad (29)$$

where T is a threshold and $\hat{x}_i$ is the restored image gray value. Fig. 3(b) shows the result of using this method with $T = 10$.

Method 2: This method simply sets λ in (12) to zero in the interior region and nonzero in the subborder region, respectively. Fig. 3(c) shows the result of using this method with $\lambda = 0.09$. In this case, D was a Laplacian operator.

Owing to checking all restored image gray values in the subborder region, Method 1 needs more computation than Method 2. However, Method 2 is very sensitive to the parameter λ, while Method 1 is not so sensitive to the parameter λ. Experimental results show that both Methods 1 and 2 reduce the ringing effect significantly by using the suboptimal blurred boundary values.

VIII. Comparisons to Other Restoration Methods

Comparing the performance of different restoration methods needs some quality measures which are difficult to define owing to the lack of knowledge about the human visual system. The word "optimal" used in the restoration techniques usually refers only to a mathematical concept, and is not related to response of the human visual system. For instance, when the blur function is ill conditioned and the SNR is low; the MMSE method improves the SNR, but the resulting image is not visually good. We believe that human objective evaluation is the best ultimate judgment. Meanwhile, the mean-square error or least square error can be used as a reference.

For comparison purposes, we give the outputs of the inverse filter, SVD pseudoinverse filter, MMSE filter, and modified MMSE filter using the Gaussian Markov random field (GMRF) model [19], [5].

A. Inverse Filter and SVD Pseudoinverse Filter

An inverse filter can be used to restore an image degraded by a space-invariant blur function with high signal-to-noise ratio. When the blur function has some singular points, an SVD pseudoinverse filter is needed; however, both filters are very sensitive to noise. This is because the noise is amplified in the same way as the signal components to be restored. The inverse filter and SVD pseudoinverse filter were applied to an image degraded by the 5 × 5 uniform blur function and quantization noise (about 40 dB SNR). The blurred and restored images are shown in Fig. 2(b) and (c), respectively. As we mentioned before, the outputs of these filters are completely overridden by the amplified noise and ringing effects.

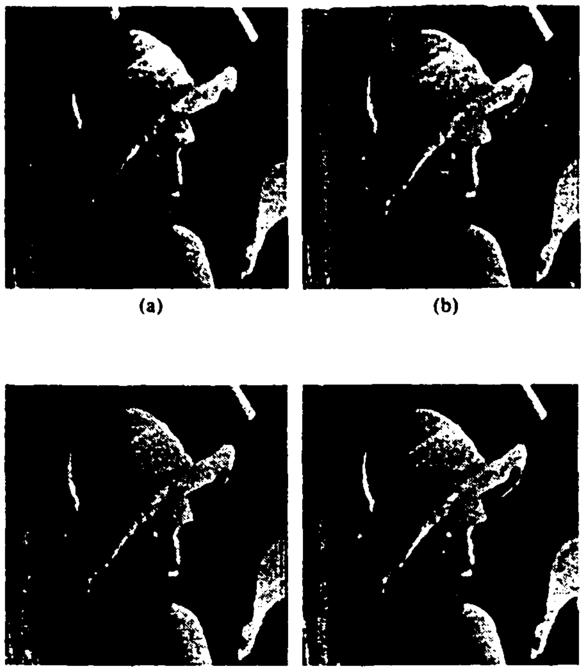

(a) (b) (c) (d)

Fig. 6. Comparison to other restoration methods. (a) Image degraded by 5 × 5 uniform blur and 20 dB SNR additive white Gaussian noise. (b) The restored image using the MMSE filter. (c) The restored image using the modified MMSE filter. (d) The restored image using our approach.

B. *MMSE and Modified MMSE Filters*

The MMSE filter is also known as the Wiener filter (in the frequency domain). Under the assumption that the original image obeys a GMRF model, the MMSE filter (or Wiener filter) can be represented in terms of the GMRF model parameters and the blur function. In our implementation of the MMSE filter, we used a known blur function, unknown noise variance, and the GMRF model parameters estimated from the blurred noisy image by a maximum likelihood (ML) method [19]. The image shown in Fig. 6(a) was degraded by 5 × 5 uniform blur function and 20 dB SNR additive white Gaussian noise. The restored image is shown in Fig. 6(b).

The modified MMSE filter in terms of the GMRF model parameters is a linear weighted combination of a Wiener filter with a smoothing operator (such as a median filter) and a pseudoinverse filter to smooth the noise and preserve the edge of the restored image simultaneously. Details of this filter can be found in [5]. We applied the modified MMSE filter to the same image used in the MMSE filter above with the same model parameters. The smoothing operator is a 9 × 9 cross shape median filter. The resulting image is shown in Fig. 6(c).

The result of our method is also shown in Fig. 6(d). The D we used in (12) was a Laplacian operator as in (13). We chose $\lambda = 0.0625$ and used 4 pixel wide blurred noisy boundaries for restoration. The total number of iterations was 20. The improvement of mean-square error between the restored image and the original image for each method is shown in Table I. In the table, the "MMSE (o)" denotes that the parameters were estimated from the original image. The restored image using "MMSE (o)" is very similar to Fig. 6(a). As we mentioned before, the comparison of the outputs of the different restoration methods is a difficult problem. The MMSE filter visually gives the worst output which has the smallest mean-square error for the MMSE (o) case. The result of our method is smoother than that of the MMSE filter. Although the output of the modified MMSE filter is smooth in flat regions, it contains some artifacts and snake effects at the edges due to using a large sized median filter.

TABLE I
MEAN-SQUARE ERROR IMPROVEMENT

Method	MMSE	MMSE (o)	Modified MMSE	Neural Network
Mean-square error	1.384 dB	2.139 dB	1.893 dB	1.682 dB

IX. Parameter Learning for Linear Image Blur Model

Apart from fine-grain parallelism, fast (and preferably automatic) adaptation of a problem-solving network to different instances of a problem is a primary motivation for using a network solution. For pattern recognition and associative memory applications, this weight training is done by distributed algorithms that optimize a distance measure between sample patterns and network responses. However, in feedback networks, general problems that involve learning higher order correlations (like the exclusive OR) or combinatorial training sets (like the Traveling Salesperson problem) are difficult to solve and may have exponential complexity. In particular, techniques for finding a compact training set do not exist.

A. *Learning Model*

For model-based approaches to "neural" problem solving, the weights of the main network are computed from the parameters of the model. The learning problem can then be solved by a parallel, distributed algorithm for estimating the model parameters from samples of the inputs and desired outputs. This algorithm can be implemented on a secondary network. An error function for this "learning" network must be constructed, which will now be problem-dependent.

For the linear shift-invariant blur model (5), the problem is that of estimating the parameters corresponding to the blur function in a $K \times K$ small window centered at each pixel. Rewrite (5) as

$$y(i,j) = z(i,j)^t h + n(i,j) \qquad i,j = 1, 2, \cdots, L \tag{30}$$

where t denotes the transpose operator and $z(i,j)$ and h are $K^2 \times 1$ vectors corresponding to original image samples in a $K \times K$ window centered at (i,j) and blur function, respectively.

For instance, for $K = 3$, we have

$$h = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_9 \end{bmatrix} = \begin{bmatrix} h(-1, -1) \\ h(-1, 0) \\ h(-1, 1) \\ \vdots \\ h(1, 1) \end{bmatrix} \tag{31}$$

and

$$z(i, j) = \begin{bmatrix} z(i, j)_1 \\ z(i, j)_2 \\ z(i, j)_3 \\ \vdots \\ z(i, j)_9 \end{bmatrix} = \begin{bmatrix} x(i-1, j-1) \\ x(i-1, j) \\ x(i-1, j+1) \\ \vdots \\ x(i+1, j+1) \end{bmatrix}. \tag{32}$$

We can use an error function for estimation of h, as in the restoration process, because the roles of data $\{x(i, j)\}$ and parameter h are simply interchanged in the learning process. Therefore, an error function is defined as

$$E = \sum_{(i,j) \in S} [y(i, j) - h^t z(i, j)]^2 \tag{33}$$

where S is a subset of $\{(i, j), i, j = 1, 2, \cdots, L\}$ and $y(i, j)$ and $z(i, j)$ are training samples taken from the degraded and original images, respectively. The network energy functions is given by

$$E = -\sum_{k=1}^{K^2} \sum_{l=1}^{K^2} w_{kl} h_k h_l - \sum_{k=1}^{K^2} \theta_k h_k \tag{34}$$

where h_k are the multilevel parameter activities and w_{kl} and θ_k are the symmetric weights and bias inputs, respectively. From (33) and (34), we get the weights and bias inputs in the familiar outer-product forms:

$$w_{kl} = -\sum_{(i,j) \in S} z(i, j)_k \, z(i, j)_l \tag{35}$$

$$\theta_k = 2 \sum_{(i,j) \in S} z(i, j)_k \, y(i, j). \tag{36}$$

A greedy, distributed neural algorithm is used for the energy minimization. This leads to a localized multilevel number representation scheme for a general network.

B. Multilevel Greedy Distributed Algorithm

For a K^2 neuron second-order network, we choose Γ discrete activities $\{f_i, i = 0, 1, \cdots, \Gamma - 1\}$ in any arbitrary range of activities (e.g., $[0, 1]$) where we shall assume without loss of generality that $f_i > f_{i-1}$ for all i. Then, between any two activities f_m and f_n for the kth neuron, we can locally and asynchronously choose the one which results in the lowest energy given the current state of the other neurons because

$$E_{h_k = f_m} - E_{h_k = f_n} = [\theta_k - \zeta_k - (f_m + f_n) w_{k,k}] \cdot [f_m - f_n] \tag{37}$$

where

$$\zeta_k = \sum_{i, i \neq k}^{K^2} w_{i,k} h_i$$

is the current weighted sum from the other neuron activities. Thus, we choose level m over n for $m > n$ if

$$\zeta_k > \theta_k - (f_m + f_n) w_{k,k}. \tag{38}$$

Some properties of this algorithm follow.

1) Convergence is assured as long as the number of levels is not decreasing with time (i.e., assured if coarse to fine).

2) Self-feedback terms are included as level-dependent bias input terms.

3) The method can be easily extended to higher order networks (e.g., based on cubic energies). Appropriate lower order level-dependent networks (like the extra bias input term above) must then be implemented.

The multilevel lowest energy decision can be implemented by using variations of feedforward min-finding networks (such as those summarized in [20]). The space and time complexity of these networks are, in general, $O(\Gamma)$ and $O(\log \Gamma)$, respectively. However, in the quadratic case, it is easy to verify from (38) that we need only implement the decision between all *neighboring* levels in the set $\{f_i\}$; this requires exactly Γ neurons with level-dependent inputs. The best activity in the set is then proportional to the sum of the Γ neuron outputs so that the time complexity for the multilevel decision can be made $O(1)$. This means that this algorithm is similar in implementation complexity (e.g., the number of problem-dependent global interconnects required) to the simple sum energy representation used in [10] and in this paper. Also, in the simple sum case, visiting the neurons for each pixel *in sequence* will result in conditional energy minimization. Otherwise, from the implementation point of view, the two methods have some properties that are complementary. For example, we have the following.

1) The simple sum method requires asynchronism in the update steps for each pixel, while the greedy method does not.

2) The level-dependent terms arise as *inputs* in the greedy method as compared to *weights* in the simple sum method.

C. Simulation Results

The greedy algorithm was used with the weights from (35) and (36) to estimate the parameters from original and blurred sample points. A 5×5 window was used with two types of blurs: uniform and Gaussian. Both real and synthetic images were used, with and without additive Gaussian noise.

TABLE II
RESULTS FOR PARAMETER LEARNING. THE NUMBER Γ OF DISCRETE ACTIVITIES IS 256 FOR ALL TESTS. *A*: ARBITRARY CHOICE OF PIXELS FROM IMAGE. *L*: PIXELS CHOSEN FROM THRESHOLDED LAPLACIAN

Image	Noise	Blur	Samples	Methods	Iterations	MSE
Synthetic		Gaussian	68	*A*	49	0.000023
Synthetic		Uniform	100	*A*	114	0.000011
Real		Uniform	50	*A*	94	0.00353
Real		Uniform	100	*L*	85	0.00014
Real	20 dB	Uniform	100	*A*	72	0.00232
Real	20 dB	Uniform	100	*L*	83	0.00054

The estimated parameters for all types of blur matrices were numerically very close to the actual values when synthetic patterns were used. The network took longest to converge with a uniform blur function. The levels chosen for the discrete activity set $\{f_i\}$ were 128–256 equally spaced points in [0, 1] with 50–100 sample points from the image. Results for various cases are summarized in Table II.

When the sample pixels were randomly chosen, the errors increased by two orders of magnitude for a real image [Fig. 2(b)] as compared to synthetic ones. This is due to the smooth nature of real images. To solve this problem, sample points were chosen so as to lie close to *edges* in the image. This was done by thresholding the Laplacian of the image. Using sample points above a certain threshold for estimation improved the errors by an order of magnitude. The results were not appreciably degraded with 20 dB noise in the samples.

X. CONCLUSION

This paper has introduced a new approach for the restoration of gray level images degraded by a shift-invariant blur function and additive noise. The restoration procedure consists of two steps: parameter estimation and image reconstruction. In order to reduce computational complexity, a practical algorithm (Algorithm 2), which has equivalent results to the original one (Algorithm 1), is developed under the assumption that the neurons are sequentially visited. The image is generated iteratively by updating the neurons representing the image gray levels via a simple sum scheme. As no matrices are inverted, the serious problem of ringing due to the ill-conditioned blur matrix H and noise overriding caused by inverse filter or pseudoinverse inverse filter are avoided by using suboptimal boundary conditions. For the case of a 2-D uniform blur plus small noise, the neural network-based approach gives high-quality images compared to some of the existing methods. We see from the experimental results that the error defined by (12) is small, while the error between the original image and the restored image is relatively large. This is because the neural network decreases energy according to (12) only. Another reason is that when the blur matrix is singular or ill conditioned, the mapping from X to Y is not one to one; therefore, the error measure (12) is not reliable anymore. In our experiments, when the window size of a uniform blur function is 3×3, the ringing effect was eliminated by using blurred noisy boundary values without any smoothing constraint. When the window size is 5×5, the ringing effect was reduced with the help of the smoothing constraint and suboptimal boundary conditions. We have also shown that a smaller secondary network can effectively be used for estimating the blur parameters; this provides a more efficient learning technique than Boltzman machine learning on the primary network.

REFERENCES

[1] H. C. Andrews and B. R. Hunt, *Digital Image Restoration*. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[2] J. W. Woods and V. K. Ingle, "Kalman filtering in two dimensions: Further results," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 188–197, Apr. 1981.

[3] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978.

[4] R. Chellappa and R. L. Kashyap, "Digital image restoration using spatial interaction models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 461–472, June 1982.

[5] H. Jinchi and R. Chellappa, "Restoration of blurred and noisy image using Gaussian Markov random field models," in *Proc. Conf. Inform. Sci. Syst.*, Princeton Univ., Princeton, NJ, 1986, pp. 34–39.

[6] N. H. Farhat, D. Psaltis, A. Prata, and E. Paek, "Optical implementation of the Hopfield model," *Appl. Opt.*, vol. 24, pp. 1469–1475, May 15, 1985.

[7] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[8] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554–2558, Apr. 1982.

[9] S.-I. Amari, "Learning patterns and pattern sequences by self-organizing nets of threshold elements," *IEEE Trans. Comput.*, vol. C-21, pp. 1197–1206, Nov. 1972.

[10] M. Takeda and J. W. Goodman, "Neural networks for computation: Number representations and programming complexity," *Appl. Opt.*, vol. 25, pp. 3033–3046, Sept. 1986.

[11] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, Sept. 1985.

[12] J. P. LaSalle, *The Stability and Control of Discrete Processes*. New York: Springer-Verlag, 1986.

[13] N. Metropolis *et al.*, "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1091, 1953.

[14] S. Kirkpatrick *et al.*, "Optimization by stimulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[15] W. K. Pratt *et al.*, "Visual discrimination of stochastic texture fields," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 796–814, Nov. 1978.

[16] J. W. Woods, J. Biemond, and A. M. Tekalp, "Boundary value problem in image restoration," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, Mar. 1985, pp. 692–695.

[17] M. M. Sondhi, "The removal of spatially invariant degradations," *Proc. IEEE*, vol. 60, pp. 842–853, July 1972.

[18] J. Biemond, J. Rieske, and J. Gerbrand, "A fast Kalman filter for images degraded by both blur and noise," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1248–1256, Oct. 1983.

[19] R. Chellappa and H. Jinchi, "A nonrecursive filter for edge preserv-

ing image restoration," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, Mar. 1985, pp. 652-655.
[20] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4-22, Apr. 1987.

Yi-Tong Zhou (S'84) received the B.S. degree in physics from the East China Normal University, Shanghai, China, and the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, in 1982 and 1983, respectively.

He is currently a Research Assistant at the Signal and Image Processing Institute, University of Southern California, Los Angeles, and is working toward the Ph.D. degree in electrical engineering. His research interests include image processing, computer vision, neural network algorithms, optical computing, and biomedical signal processing. He has published about a dozen technical papers in these areas.

Rama Chellappa (S'78-M'79-SM'83), for a photograph and biography, see this issue, p. 1066.

Aseem Vaid was born in Jammu, India, on January 8, 1963. He received the B.Tech. degree in electrical engineering in May 1985 from the Indian Institute of Technology, New Delhi.

Currently he is working on the Ph.D. degree at the University of Southern California, Los Angeles. His research interests are in neural networks and optical computing.

B. Keith Jenkins (M'85) received the B.S. degree in applied physics from the California Institute of Technology, Pasadena, in 1977, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, in 1979 and 1984, respectively.

He was employed at Hughes Aircraft Company, El Segundo, CA, from 1977 to 1979 where he worked on holography for use in head-up displays. From 1984 to 1987 he was a Research Assistant Professor in the Department of Electrical Engineering, University of Southern California, where he presently is Assistant Professor of Electrical Engineering. He has also participated in advisory panels to government and industry, and has been a consultant to JPL, TRW, and Odetics. His research has included work in the areas of optical digital computing, neural networks and their optical implementation, learning algorithms, optical interconnection networks, parallel computation models and complexity, computer-generated holography, and optical 3-D position sensing systems.

Dr. Jenkins is a member of the Optical Society of America and the Association for Computing Machinery. He was awarded The Northrop Assistant Professor of Engineering at USC in 1987, and is a recipient of the 1988 NSF Presidential Young Investigator Award.

Implementation Considerations of a Subtracting Incoherent Optical Neuron

• C. H. Wang and B. K. Jenkins

Signal and Image Processing Institute, Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089-0272

ABSTRACT

The incoherent optical neuron (ION) subtracts inhibitory inputs from excitatory inputs optically by utilizing separate device responses. Those factors that affect the operation of the ION are discussed here, such as nonlinearity of the inhibitory element, input noise, device noise, system noise and crosstalk. A computer simulation of these effects is performed on a version of Grossberg's on-center off-surround competitive neural network.

1 Introduction

The need to process positive and negative signals optically in optical neural network has been pursued in the past few years. Existing techniques such as intensity bias [1] or weight bias method suffer from input dependent bias or thresholds that must vary from neuron to neuron. A technique described by Te Kolste and Guest [2] eliminites most of these drawbacks in the special case of fully connected networks.

The *incoherent optical neuron* (ION) [3, 4] model uses separate device responses for inhibitory and excitatory inputs. This is modeled after the biological neuron that processes the excitatory and inhibitory signals by different mechanisms (e.g. chemical-selected receptors and ion-selected gate channels) [5, 6, 7, 8]. By using this architecture, we can realize general optical neuron units with thresholding.

The ION comprises two elements: an inhibitory (I) element and a nonlinear output (N) element. The inhibitory element provides inversion of the sum of the inhibitory signals; the nonlinear element operates on the excitatory signals, the inhibitory element output, and an optical bias to produce the output. The inhibitory element is linear; the nonlinear threshold of the neuron is provided entirely by the output nonlinear device. Fig. 1(a) and 1(c) shows the characteristic curve of the I and N elements respectively. The structure of the ION model is illustrated in Fig. 1(d). The input/output relationships for the normalized I and N elements respectively, are given by:

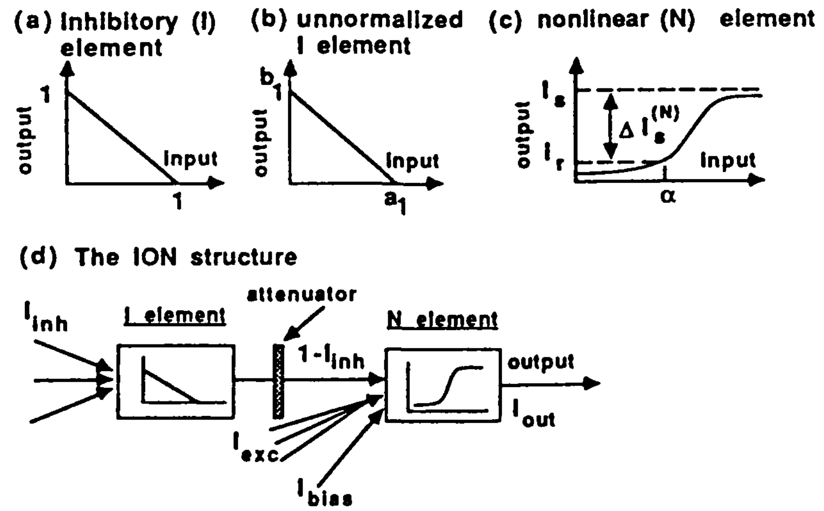


Fig. 1 The ION: (a)-(c) its components, and (d) its structure.

• Proc. IEEE International Conference on Neural Networks, San Diego, CA, July 1988. vol. II, 403-410

$$I_{out}^{(I)} = \hat{I}_{inh} = 1 - I_{inh} \tag{1}$$

$$I_{out}^{(N)} = \psi(\hat{I}_{inh} + I_{exc} + I_{bias} - \alpha) \tag{2}$$

where I_{inh} and I_{exc} represent the total inhibitory and excitatory inputs, I_{bias} is the bias term for the N element, which can be varied to change the threshold, and α is the offset of the characteristic curve of the N element. $\psi(\cdot)$ represents the output nonlinear function of the N element. If we choose I_{bias} to be $\alpha - 1$, the output of the N element is

$$I_{out}^{(N)} = \psi(I_{exc} - I_{inh}) \tag{3}$$

which is the desired subtraction. In general, the I element will not be normalized (Fig 1(b)), in which case the offset and slope of its response can be adjusted using I_{bias} and an attenuating element (ND filter), respectively. The unnormalized I element must have gain greater than or equal to 1. A positive threshold (θ) can be implemented by lowering the bias term by the same amount θ. Similarly, a negative threshold is realized by increasing the bias term by θ.

The ION model can be implemented using separate devices for the I and N elements (heterogeneous case), or by using a single device with a nonmonotonic response to implement both elements (homogeneous case). Possible devices include bistable optical arrays [9, 10, 11, 12] and SLMs such as liquid crystal light valves (LCLV) [13]. A single Huges liquid crystal light valve can be used to implement both elements (Fig. 2).

Several factors that affect the realization of a neural network based on the ION concept, are examined here. These include deviation from linearity within the inhibitory element, residual noise of the optical device, input noise, drift of the operation point of the device, and system noise. A noise model for the ION is proposed and a computer simulation of these effects on a version of Grossberg's on-center off-surround type network [14] is performed.

2 Factors that Affect the ION Operation

2.1 Nonlinearity in I Element

In order to perform subtraction correctly, we need a linear I element. Fig. 2 shows the typical input output characteristic curve of the LCLV [15], which is nonlinear in the inversion region. In this region, the characteristic curve can be modeled as

$$I_{out}^{(I)} = 1 - I_{inh} - E_r(I_{inh}) \tag{4}$$

where $E_r(I_{inh})$ denotes the error term, which can be treated as an input dependent deterministic noise. If the transfer curve is time varying, then it can be treated as temporally correlated random noise.

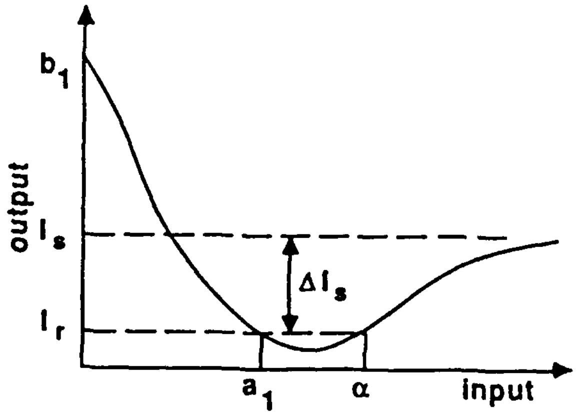


Fig. 2 Characteristic of a Hughes twisted-nematic liquid crystal light valve. The negative slope region can implement the I element, and the positive slope region, with appropriate input optical bias, can implement the N element.

2.2 Noise

Here we use "noise" to mean "any undesired signals", including perturbation of the operating point of the device, non-uniformity of the device, variation in operating characteristics from device to device due to production variation, environmental effects etc. Some of these effects are global (they affect all neuron units on a device identically), others are localized (each neuron unit behaves differently; the noise on neighboring neuron units on a device may be independent or correlated, depending on the source of the noise). Both temporal and spatial characteristics of the noise need to be included. The effect of noise on an additive lateral inhibitory network was discussed by Stirk, Rovnyak and Athale [16]. Here, we construct a noise model for the ION by considering the origin and impact of the noise sources.

The possible noise sources in the ION model can be classified into four categories: input noise, device noise, system noise and coupling noise. The input noise includes environmental background noise, residual output of the optical devices etc. Essentially, they are not zero mean and vary slowly with time. The device noise is mainly caused by uncertainty in the device's characteristics, for example drift of the operating point and variation of gain, due to temperature or other effects. The system noise has global effect on all neuron units on an optical device and includes fluctuations in the optical source. Finally, the coupling noise (crosstalk) is due to poor isolation between the optical neuron units, crosstalk from the interconnection network, and imperfect learning. As noted in [16], alignment inaccuracies and imperfect focussing and collimating optics also cause localized crosstalk. Coupling noise is signal dependent.

2.3 Noise Model for the ION

Device Input Noise

Let the environmental background noise for the I and N elements be denoted by $N_b^{(I)}$ and $N_b^{(N)}$ respectively. The total residual output noise, caused by the optical devices to the input of an incoherent optical neuron, is $N_r = \sum_{j=1}^{N_{in}} W_{ij} I_r / N_{out}$, which is weight-dependent and varies slowly with time due to learning. W_{ij} is the interconnection strength from neuron j to neuron i. I_r is the residual output of the optical device (Fig. 1(c)). N_{in} and N_{out} denote the fan-in and fan-out of the optical neuron unit respectively. Perturbation of the weights can be treated as an input dependent noise source as $N_w = \sum \Delta W_{ij} \cdot x_j$, where each ΔW_{ij} is independent. For the interconnection network, imperfect learning of the weights, nonuniformity of the weights, residual weights after reprogramming and perturbation of the reference beam intensity will cause weight noise. Then the output of the ION for the case of normalized characteristics is

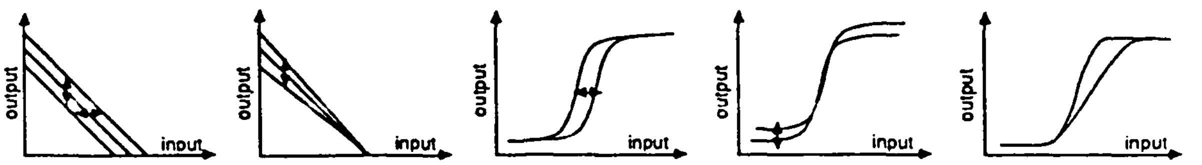
$$I_{out} = \psi\{[1 - (I_{inh} + N_b^{(I)} + N_r^{(I)} + N_w^{(I)})] + [I_{exc} + N_b^{(N)} + N_r^{(N)} + N_w^{(N)}] + (\alpha - 1) - \alpha\} \qquad (5)$$

If the background noise is space invariant and the I and N element have the same device area, the terms $N_b^{(I)}$ and $N_b^{(N)}$ will cancel out. The residual noise terms $N_r^{(I)}$ and $N_r^{(N)}$, and weight noise $N_w^{(I)}$ and $N_w^{(N)}$ generally do not cancel.

Device Noise

There are two possible noise sources in the I element, as illustrated in Fig. 3(a) and (b): shift (drift) and gain variation in the device characteristics, which are denoted as $N_d^{(I)}$ and $N_g^{(I)}$ respectively. For the output N element, the gain variation (Fig. 3(e)) only modifies the nonlinearity of the element N. If this gain variation is a slowly varying effect, it will have little effect on the dynamic behavior of the network; so for the N element we only consider the drift effect. Let's denote it by $N_d^{(N)}$. Two different drifts in the N element are possible, horizontal drift ($N_{dh}^{(N)}$) (Fig. 3(c)) and vertical drift ($N_{dv}^{(N)}$) (Fig 3(d)). The vertical drift of one neuron unit becomes an additive noise at the input of the next neuron unit, and so will be approximated by including it in the residual noise term above. The horizontal drift has the same effect as a perturbation in the bias term, denoted by N_{bp}.

If the gain variation is small, the output of the I element can be expressed as $(1 + N_g) - (1 + N_g) I_{inh}$, where N_g denotes the gain noise.



(a) drift (vertical/horizontal) of the I element. (b) gain variation effect of the I element. (c) horizontal drift of the N element or variation on the bias. (d) vertical drift of the N element. (e) gain variation effect of the N element.

Fig. 3 Modeling the device noise of an incoherent optical neuron.

System Noise

System noise has a global effect on the ION. If it is caused by an uncertainty in the optical source, it causes a variation in the characteristic curve of the device and a perturbation in the bias term. In the case of an LCLV, a perturbation in the reading beam intensity produces a gain variation in the I element and a combination of gain variation and horizontal drift in the N element (or equivalently, essentially a re-scaling of its input axis). Gain variation of the I element was discussed above. The difference is that the device gain variation is local, ie. it varies from one neuron unit to another on a given device, while the variation in gain due to system noise is global.

The device noise and system noise can be modeled as

$$I_{out} = \psi\{(1 + N_g^{(I)})[1 + N_d^{(I)} - I_{inh}] + I_{exc} + N_{dh}^{(N)} + (\alpha - 1 + N_{bp}) - \alpha\} \tag{6}$$

Crosstalk

Crosstalk can be caused by the physical construction of the interconnection network (e.g., coupling between different holograms, diffraction in the detection (neuron unit inputs) plane, inaccurate alignment and focussing). It can also be caused by imperfect learning or reprogramming of the synaptic weights, where the perturbation of different weights are correlated. In general, crosstalk is signal dependent and varies from one neuron unit to another on a given device. It can be modeled as an input noise to the I and N elements. It is excluded from our current simulations because it is signal dependent.

Based on the above discussion, these noise terms can be grouped into additive (N_I^+) and multiplicative (N_I^*) noise of the I element and additive noise (N_N^+) of the output element N. The general noise model of the ION can be written as

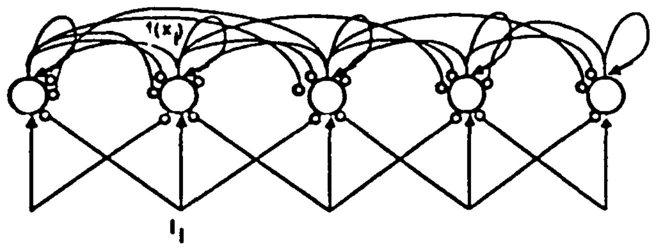
$$I_{out} = \psi\{(1 + N_I^*)[1 - I_{inh} + N_I^+] + I_{exc} + N_N^+ - 1\} \tag{7}$$

where N_I^+ is the sum of the drift noise ($N_d^{(I)}$), background noise ($N_b^{(I)}$), residual noise ($N_r^{(I)}$), weight noise ($N_w^{(I)}$) and crosstalk noise ($N_c^{(I)}$); N_I^* is the gain noise of I element; and N_N^+ is the sum of the background ($N_b^{(N)}$) and residual input noise, horizontal shift noise ($N_{dh}^{(N)}$), weight noise and crosstalk noise of the N element, and bias noise (N_{bp}).

3 Computer Simulation

3.1 Compensation for the Nonlinearity of the I Element

To assess the effect of imperfect device responses for the I element, we have performed simulations on a variant of Grossberg's on-center off-surround competitive network [14] for edge detection (Fig. 4). The network contains 30 inner-product type neurons connected in a ring structure with input and lateral on-center off-surround connections. Fig. 5 shows several modeled nonlinear characteristic curves for the I element; these approximate the normalized response of a Hughes liquid-crystal light valve. An attenuator (neutral density filter) can be placed in front of the I element to reduce the overall gain (by effectively re-scaling the input axis) to bring it closer to the ideal response. Fig. 6 shows computer simulations of the network responses based on nonlinear curve #2 for different input attenuations and input levels. As shown here, the attenuation has a tolerance of approximately ±20%. For extremely nonlinear responses we expect an input bias and a limited region of



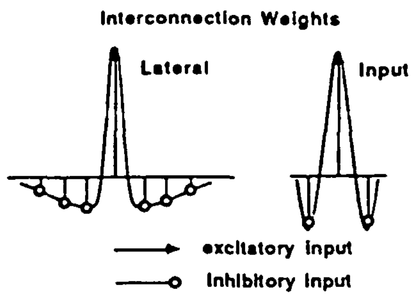


Fig. 4 On-center off-surround competitive neural network.

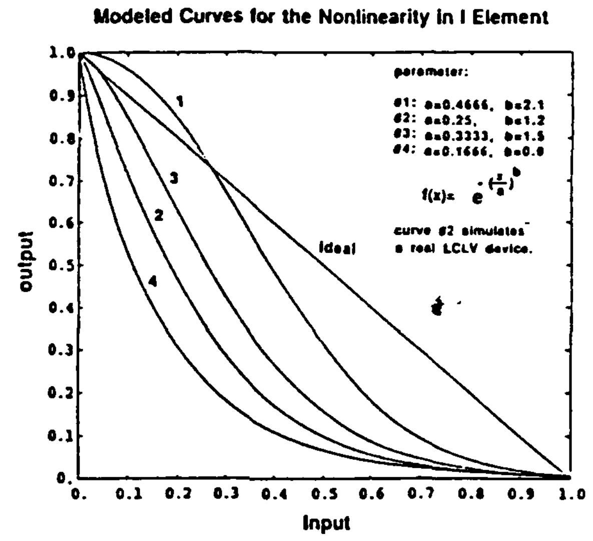


Fig. 5 Four curves used for simulating the effect of nonlinearities in the I element.

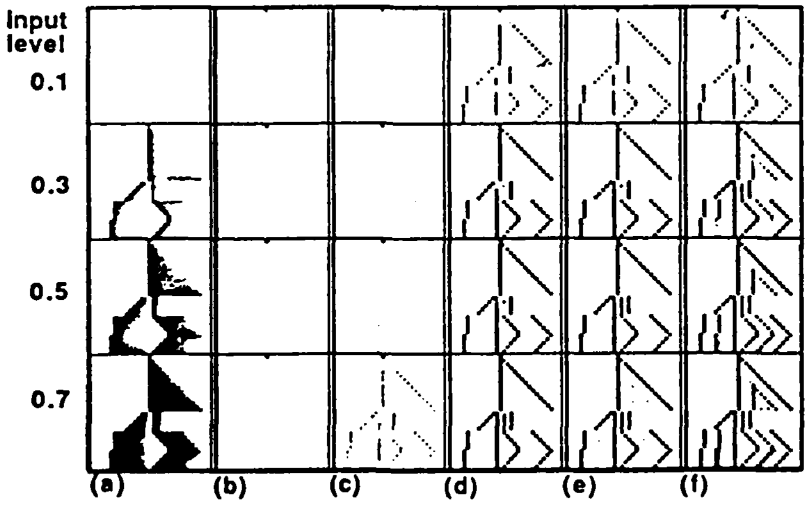


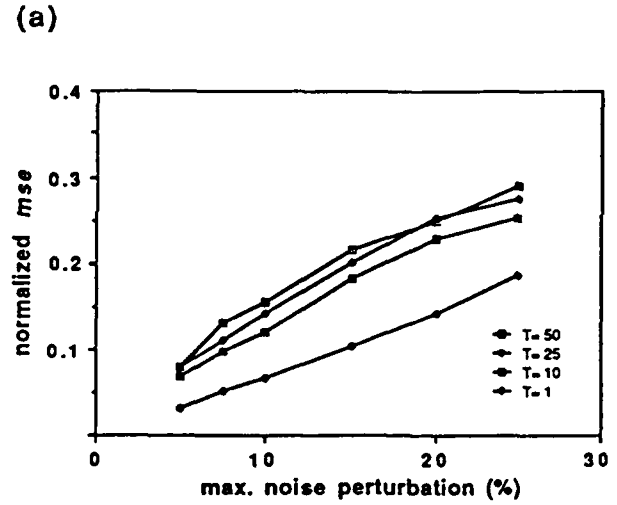
Fig. 6 Network responses for different attenuation factors, S_1, at the input to the nonlinear inhibitory element.
a) input patterns. b) $S_1 = 1.0$ (no attenuation), c) $S_1 = 1.5$, d) $S_1 = 2.5$, e) $S_1 = 3.0$, f) $S_1 = 3.5$. The ideal output is essentially identical to (d).

operation to provide a sufficiently linear response. In our simulations we used an attenuator but no input bias to the I element; the region of operation for these curves was seen to extend over most of the input range of the device [3,4].

3.2 Noise Effect

We use the same network to test effect of noise (of course the results are actually network dependent) to get an idea of the noise imunity and robustness to physical imperfections of the ION model. In the computer simulation, each of the three noise sources in Eq. (7) are assumed independently distributed Gaussian with zero mean. We define the maximum perturbation, p, of the noise source as twice the standard deviation, expressed as a percentage of the input signal level. A normalized mean square error (*nmse*) is used to measure the acceptability of the result. Although it is not a perfect measure, a *nmse* less than 0.1-0.15 generally looks acceptable for the network response for our input test pattern.

Fig. 7(a) shows the *nmse* vs. percentage of maximum noise perturbation for the input level of 0.7 and for noise that is correlated over different time periods T. The noise sources for each neuron are assumed independent and identically distributed (*iid*). Each noise source is temporally correlated with its previous values, as given by $N(t+1) = \sum_{i=1}^{T} h_i \cdot N(t+1-i)$. The correlation coefficients h_i decrease linearly with i (to $h_T = 0$). In Fig. 7(a), all three noise sources in our model are present and have the same variance. If the acceptance *nmse* criterion is 0.15, a perturbation of ±10% on each noise source yields an acceptable result in all cases. For $T = 50$, the *nmse* increases as the input level and noise variance increase as shown in Fig. 7(b). The network responses are shown in Fig. 8 for temporally correlated noise with perturbation of ±10%.



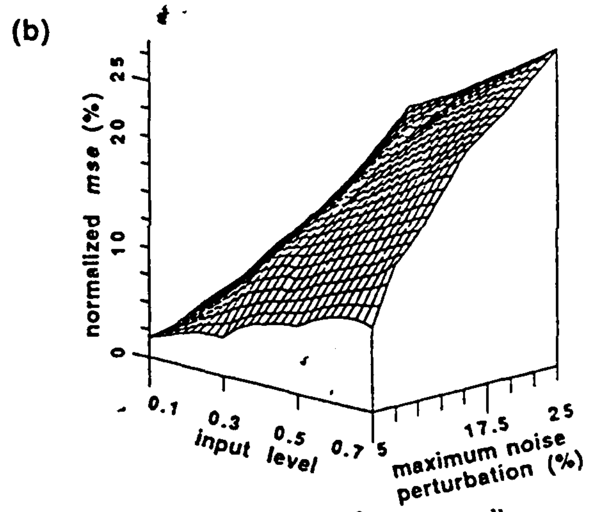


Fig. 7 Normalized mean square error (*nmse*) measure of the network response for temporally correlated noise. Three noise sources, N_I^+, N_I^*, and N_N^+, are simulated.

a) Normalized mean square error of the net output vs. maximum noise perturbation p for correlation periods (T) ranging from 1 to 50. The input level is 0.7.

b) Output *nmse* plot for different noise perturbation and input levels (T=50).

In some cases, the noise is spatially correlated. We simulated the network with spatially correlated noise. The spatial correlation is assumed to have a Gaussian profile. Fig 9(a) and (b) are the responses for a spatial correlation range of 5 and 13 respectively, while Fig 9(c) and (d) show the responses for spatially *and* temporally correlated noise.

Drift of the device characteristic is a global effect. Fig 10 simulates slowly varying and quickly varying drift on this network. Fig. 11 shows the effect of local gain variation that is spatially correlated. A ±25% perturbation in drift is apparently acceptable, and a ±15 – 20% perturbation in gain is acceptable.

4 Discussions and Conclusions

We have summarized sources of noise for the ION and proposed a noise model. From the result of the computer simulation, it seems that the example network performs much better for quickly varying (ie. temporally

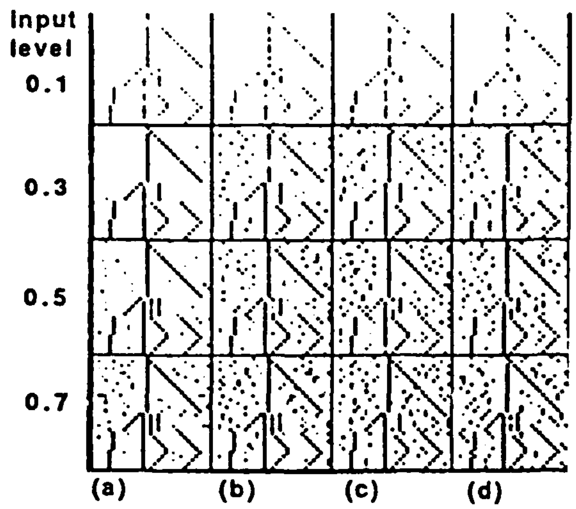


Fig. 8 The network response for temporally correlated noise sources. Three noise are simulated with the same maximum noise perturbation of ±10%. T is the correlation period of the noise and *nmse* is the normalized mean square error of the output. Here, the given value of *nmse* corresponds to the maximum input level case.

a) T=1, *nmse* = 0.07, b) T=10, *nmse* = 0.12, c) T=25, *nmse* = 0.14, d) T=50, *nmse* = 0.16.

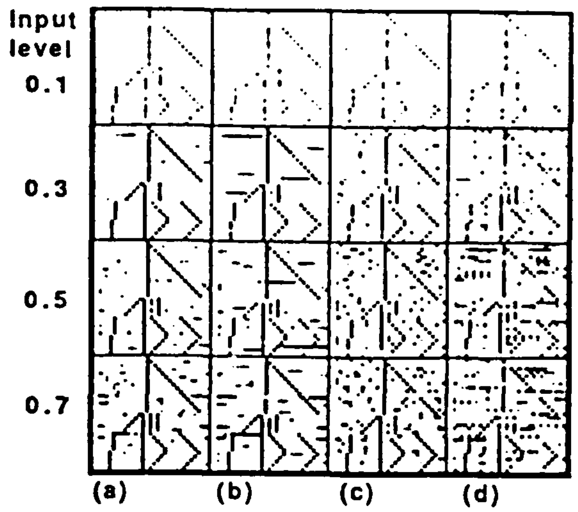


Fig. 9 Simulation result of spatially and temporally correlated noise. *sc* is the spatial correlation range. Three noise sources are simulated simulateneously with p = ±10% and correlation period T.

a)-b) only spatially correlated noise with a) *sc* = 3, *nmse* = 0.08, b) *sc* = 13, *nmse* = 0.13, c)-d) are the result of spatially and temporally correlated noise (T=25) with c) *sc* = 3, *nmse* = 0.14, d) *sc* = 13, *nmse* = 0.18.

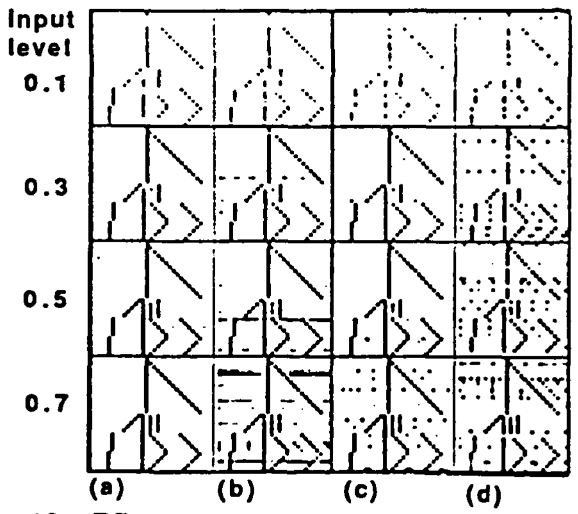


Fig. 10 Effect of device drift in the ION. p is the maximum perturbation of the noise source. T is the temporal correlation period of the drift. The drift effect is uniform over all neuron units. a) & b) simulate high frequency drift (T=1) with a) p = ±10%, *nmse* = 0.02, b) p = ±25%, *nmse* = 0.09, c)-d) are the low frequency drift (T=50) with c) p = ±10%, *nmse* = 0.07, d) p = ±25%, *nmse* = 0.11.

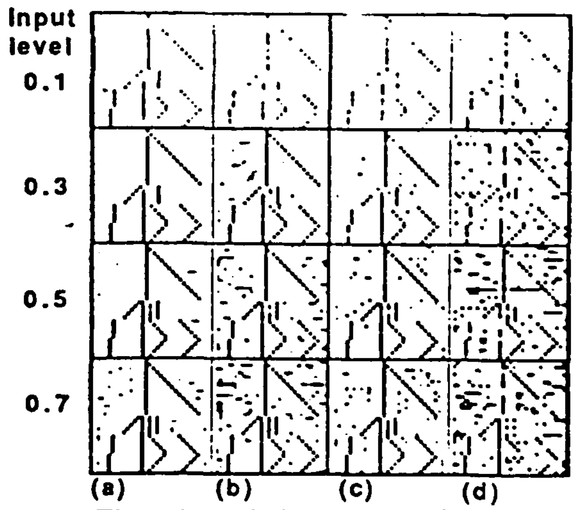


Fig. 11 The gain variation effect of the ION. *sc* and p are the correlation range and maximum perturbation percentage of the noise source. a)-b) high frequency gain variation with a) T=1, *sc* = 3, , p = ±10%, *nmse* = 0.04, b) T=1, *sc* = 3, p = ±25%, *nmse* = 0.11. c)-d) low frequency variation with c) T=25, *sc* = 9, p = ±10%, *nmse* = 0.09, d) T=25, *sc* = 9, p = ±25%, *nmse* = 0.18.

uncorrelated) noise than for temporally correlated (more slowly varying) noise. Due to the static input pattern and the competitive nature of the network, once the noise term has survived a number of iterations, then it will continue to get stronger and will not die out. We speculate that if the input to the network is time varying, then the slowly varying noise source is effectively an offset response of the network and might be adaptively overcome by the network, while the quickly varying noise interacts with the input patterns and is more difficult to compensate.

For noise that is correlated, we have found that the qualitative effect of each of the three noise sources (additive inhibitory, multiplicative inhibitory, and additive excitatory) on the output of the net is essentially the same. Since one of the noise terms, N_N^+, is the same for a conventional neuron implementation as for the ION, it appears that an ION implementation is not significantly different from a conventional neuron implementation in terms of immunity to noise and device imperfections, for a given technology. We also see that the output is affected primarily by the *variance* of the noise and by the degree of *spatial and temporal correlation*, but apparently *not* by the source of the noise. We conjecture that this result is not peculiar to the ION model, but is true of other neuron implementations as well.

References

[1] A. F. Gmitro and G. R. Gindi, "Optical neurocomputer for implementation of the Marr-Poggio stereo algorithm" *Proc. of IEEE First Int. Conf. on Neural Network*, San Diego, vol III, pp599-606, 1987.

[2] R. D. Te Kolste and C. C. Guest, "Optical competitive neural network with optical feedback", *Proc. of IEEE First Int. Conf. on Neural Network*, San Diego, vol III, pp625-629, 1987.

[3] B. K. Jenkins and C. H. Wang, "Requirements for an incoherent optical neuron that subtracts", *J. of Opt. Soci. Am. (A)*, vol 4, No. 13, 127A, paper PD6, Dec. 1987.

[4] B. K. Jenkins and C. H. Wang, "Model for an incoherent optical neuron that subtracts", submitted to *Optics Letters*, 1988.

[5] M. Wang and A. Freeman, "Neural function", *Little, Brown and Company* press, 1987.

[6] C. F. Stevens, "The neuron", *Scientic American*, pp55-65, Aug. 1979.

[7] G. M. Shepherd, "Microcircuits in the nervous system", *Scientic American*, pp93-103, Feb. 1978.

[8] R. D. Keynes, "Ion channels in the nerve-cell membrane", *Scientic American*, pp126-135, March 1979.

[9] A. C. Walker, "Application of bistable optical logic gate arrays to all-optical digital parallel processing", *Applied Optics*, vol. 25, No. 10, pp1578-1585, May 1986.

[10] S. D. Smith, "Optical bistability, photonic logic, and optical computation", *Applied Optics*, vol. 25, No. 10, pp1550-1564, May 1986.

[11] S. D. Smith, A. C. Walker, et. al., "Cascadable digital optical logic circuit elements in the visible and infrared: demonstration of some first all-optical circuits", *Applied Optics*, vol. 25, No. 10, pp1586-1593, May 1986.

[12] F. A. P. Tooley, S. D. Smith, and C. T. Seaton, "High gain signal amplification in an InSb transphasor at 77 K", *Appl. Phys. Lett.*, vol 43, pp9-, 1983.

[13] W. P. Bleha et. al., "Application of the liquid crystal light valve to real-time optical data processing", *Optical Engineering*, vol 17, No. 4, pp371-384, July/Aug., 1978.

[14] S. A. Ellias and S. Grossberg, "Pattern formation, contrast control and oscillations in short term memory of shunting on-center off-surround networks", *Biol. Cybernetics*, vol 20, pp69-98, 1975.

[15] B. K. Jenkins, A. A. Sawchuk, T. C. Strand et. al., "Sequential optical logic implementation", *Appl. Optics*, vol 23, No. 19, pp3455-3464, 1 Oct. 1984.

[16] C. W. Stirk, S. M. Rovnyak and R. A. Athale, "Effects of system noise on an optical implementation of an additive lateral inhibitory network", *IEEE first International Conf. on Neural Network*, San Diego, vol III, pp615-624, 1987.

Superposition in Optical Computing

B. Keith Jenkins
Signal and Image Processing Institute MC-0272
University of Southern California, Los Angeles, California 90089-0272

and

C. Lee Giles
Air Force Office of Scientific Research/NE
Bolling AFB, D.C. 20332-6448

ABSTRACT

The design of an optical computer must be based on the characteristics of optics and optical technology, and not on those of electronic technology. The property of optical superposition is considered and the implications it has in the design of computing systems is discussed. It can be exploited in the implementation of optical gates, interconnections, and shared memory.

INTRODUCTION

Fundamental differences in the properties of electrons and photons provide for expected differences in computational systems based on these elements. Some, such as the relative ease with which optics can implement regular, massively parallel interconnections are well known. In this paper we examine how the property of superposition of optical signals in a linear medium can be exploited in building an optical or hybrid optical/electronic computer. This property enables many optical signals to pass through the same point in space at the same time without causing mutual interference or crosstalk. Since electrons do not have this property, this helps to shed more light on the role that optics could play in computing. We will separately consider the use of this property in interconnections, gates, and memory.

INTERCONNECTIONS

A technique for implementing optical interconnections from one 2-D array to another (or within the same array) has been described [Jenkins et al, 1984]. It utilizes two holograms in succession (Fig. 1). The holograms can be generated by a computer plotting device. The idea is to define a finite number, M, of distinct interconnection patterns, and then assemble the interconnecting network using only these M patterns. The second hologram of Fig. 1 consists of an array of facets, one for each of the M interconnection patterns. The first hologram contains one facet for each input node, and serves to address the appropriate patterns in the second hologram.

It is the superposition property that makes this interesting. Note that many different signal beams can pass through the same facet of the second hologram at the same time without causing mutual interference. (All of these signals merely get shifted in the same direction and by the same amount.) This feature decreases the complexity of both holograms -- The first because it only has to address M facets, the second hologram because it only has M facets. Let N be the number of nodes in the input and output arrays. The complexity (number of resolvable spots) of each hologram can be shown to be proportional to NM, with the proportionality constant being approximately 25 [Jenkins et al., 1984].

Using this as a model for interconnections in parallel computing, a comparison can be made between the complexity of these optical interconnections with those of electronic VLSI for various interconnection networks. Results of this have been given in [Giles and Jenkins, 1986]. It is found that in general the optical interconnections have an equal or lower space complexity than electronic interconnections, with the difference becoming more pronounced as the connectivity increases.

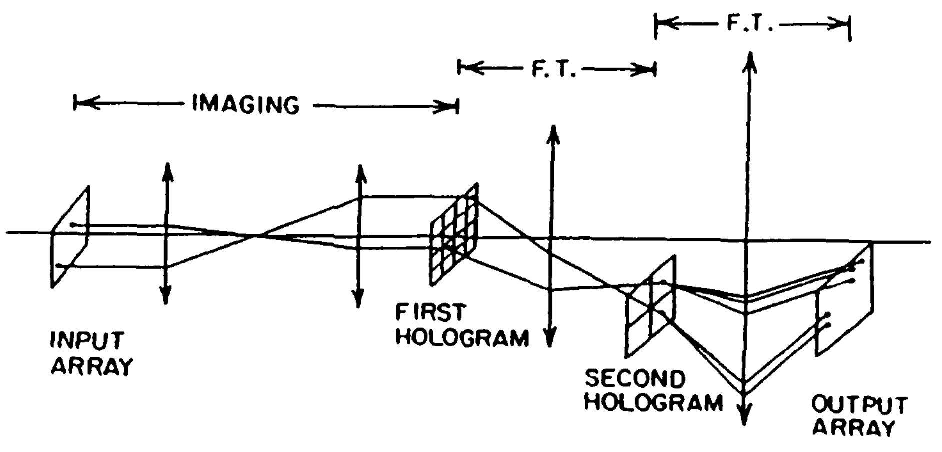


Fig. 1. Optical holographic system for interconnections.

SHARED MEMORY

The same superposition principle can be applied to memory cells, where many optical beams can read the same memory location simultaneously. This concept could be useful in building a parallel shared memory machine.

For this concept, we first consider abstract models of parallel computation based on shared memories. The reason for this approach is to abstract out inherent limitations of electronic technology (such as limited interconnection capability); in designing an architecture one would adapt the abstract model to the limitations of optical systems. These shared memory models are basically a paralleliza-tion of the Random Access Machine.

The Random Access Machine (RAM) model [Aho, Hopcroft, and Ullman, 1974] is a model of sequential computation, similar to but less primitive than the Turing machine. The RAM model is a one-accumulator computer in which the instructions are not allowed to modify themselves. A RAM consists of a read-only input tape, a write-only output tape, a program and a memory. The time on the RAM is bounded above by a polynomial function of time on the TM. The program of a RAM is not stored in memory and is unmodifiable. The RAM instruction set is is small and consists of operations such as store, add, subtract, and jump if greater than zero; indirect addresses are permitted. A common RAM model is the uniform cost one, which assumes that each RAM instruction requires one unit of time and each register one unit of space.

Shared memory models are based on global memories and are differentiated by their accessibility to memory. In Fig. 2 we see a typical shared memory model where individual processing elements (PE's) have variable simultaneous access to an individual memory cell. Each PE can access any cell of the global memory in unit time. In addition, many PE's can access many different cells of the global memory simultaneously. In the models we discuss, each PE is a slightly modified RAM without the input and output tapes, and with a modified instruction set to permit access to the global memory. A separate input for the machine is provided. A given processor can generally not access the local memory of other processors.

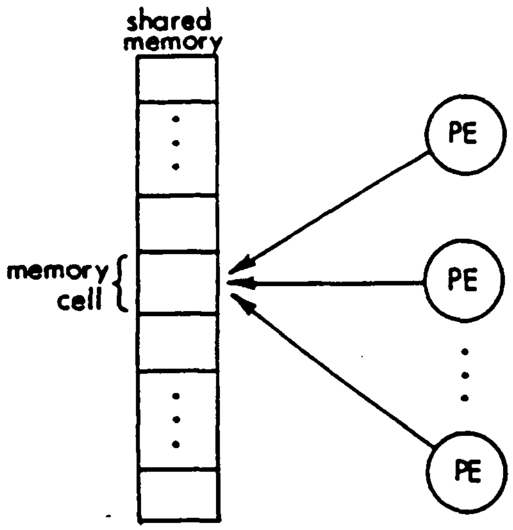


Fig. 2. Conceptual diagram of shared memory models.

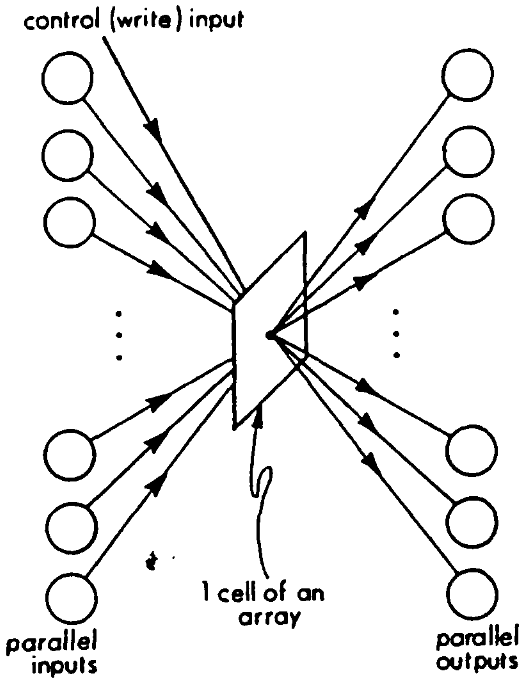


Fig. 3. One memory cell of an array, showing multiple optical beams providing contention-free read access.

The various shared memory models differ primarily in whether they allow simultaneous reads and/or writes to the *same* memory cell. The PRAC, parallel random access computer [Lev, Pippenger and Valiant, 1981] does not allow simultaneous reading or writing to an individual memory cell. The PRAM, parallel random access machine, [Fortune and Wyllie, 1978] permits simultaneous reads but not simultaneous writes to an individual memory cell. The WRAM, parallel write random access machine, denotes a variety of models that permit simultaneous reads and certain writes, but differ in how the write conflicts are resolved. For example, a model by Shiloach and Vishkin (1981) allows a simultaneous write only if all processors are trying to write the same value. The paracomputer [Schwartz, 1980] has simultaneous writes but only "some" of all the information written to the cell is recorded. The models represent a hierarchy of time complexity given by

$$T^{\mathrm{PRAC}} \geq T^{\mathrm{PRAM}} \geq T^{\mathrm{WRAM}}$$

where T is the minimum number of parallel time steps required to execute an algorithm on each model. More detailed comparisons are dependent on the algorithm [Borodin and Hopcroft, 1985].

In general, none of these shared memory are physically realizable because of actual fan-in limitations. As an electronic example, the ultracomputer [Schwartz, 1980] is an architectural manifestation of the paracomputer that uses a hardwired Omega network between the PE's and memories; it simulates the paracomputer within a time penalty of $O(\log^2 n)$. The current IBM RP3 project is a continuation of the (initial) work on the ultracomputer.

Optical systems could in principle be used to implement this parallel memory read capability. As a simple example, a single 1-bit memory cell can be represented by one pixel of a 1-D or 2-D array; the bit could be represented by the state (opaque or transparent) of the memory cell. Many optical beams can simultaneously read the contents of this memory cell without contention (Fig. 3). In addition to this an interconnection network is needed between the PE's and the memory, that can allow any PE to communicate with any memory cell, preferably in one step, and with no contention. A regular crossbar is not sufficient for this because fan-in to a given memory cell must be allowed. Figure 4 shows a conceptual block diagram of a system based on the PRAM model; here the memory array operates in reflection instead of transmission. The fan-in required of the interconnection network is also depicted in the figure.

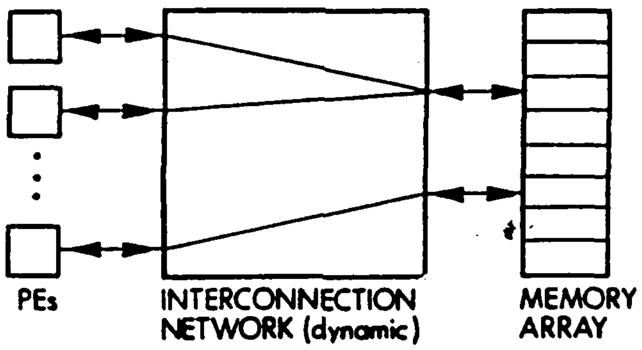


Fig. 4. Block diagram of an optical architecture based on parallel RAM models.

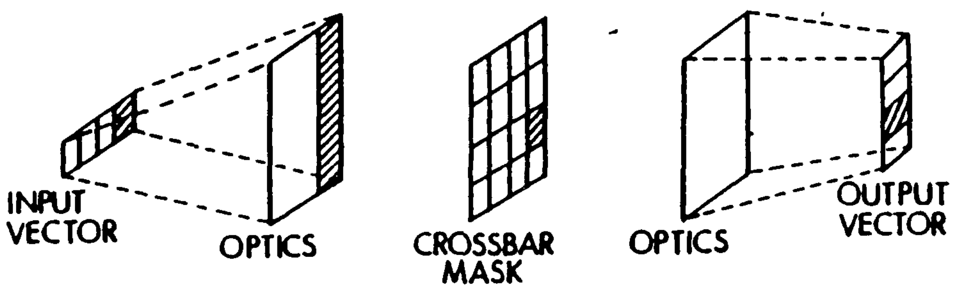


Fig. 5. Example of an optical crossbar interconnection network.

Optical systems can potentially implement crossbars that also allow this fan-in. Several optical crossbar designs discussed in [Sawchuk, et al., 1986] exhibit fan-in capability. An example is the optical crossbar shown schematically in Fig. 5; it is based on earlier work on optical matrix-vector multipliers. The 1-D array on the left could be optical sources (LED's or laser diodes) or just the location of optical signals entering from previous components. An optical system spreads the light from each input source into a vertical column that illuminates the crossbar mask. Following the crossbar mask, a set of optics collects the light transmitted by each row of the mask onto one element of the output array. The states of the pixels in the crossbar mask (transparent or opaque) determine the state of the crossbar switch. Multiple transparent pixels in a column provide fanout; multiple transparent pixels in a row provide fan-in. Many optical reconfigurable network designs are possible, and provide tradeoffs in performance parameters such as bandwidth, reconfiguration time, maximum number of lines, hardware requirements, etc. Unfortunately, most simple optical crossbars will be limited in size to approximately 256 x 256 (Sawchuk, et al., 1986). We are currently considering variants of this technique to increase the number of elements. Possibilities include using a multistage but nonblocking interconnection network (e.g. Clos), a hierarchy of crossbars, and/or a memory hierarchy.

GATES

Since the superposition property of optics only applies in linear media, it cannot in general be used for gates, which of course are inherently nonlinear. However, for important special cases superposition can allow many optical gates to be replaced with one optical switch.

Consider again the situation depicted in Fig. 3, with the aperture being used as a switch or relay. The control beam opens or closes the relay; when the relay is closed (i.e., aperture is transparent), many optical signal beams can independently pass through the relay. If b represents the control beam and a_i the signal beams, this in effect computes $b \cdot a_i$ or $\bar{b} \cdot a_i$, depending on which state of b closes the relay, where $\cdot$ denotes the AND operation (Fig. 6).

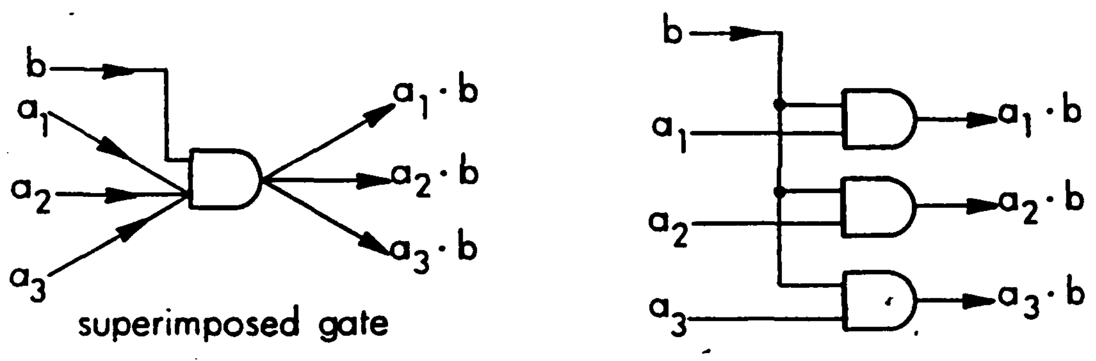


Fig. 6. One optical relay or superimposed gate versus individual gates with a common input.

Using this concept, a set of gates with a common input in a single-instruction multiple-data (SIMD) machine can be replaced with one optical switch or "superimposed gate". An example of this is in the control signals; instead of broadcasting each instruction or control bit to all PE's, a fan-in from all PE's to a common control switch is performed. Thus, for I control bits per instruction word, I superimposed gates could replace NI gates (I per PE). Since for optical or hybrid systems we expect $N \gg I$, this can be a substantial reduction. Fig. 7 shows an example of how this can be incorporated into fixed optical interconnections (such as those of Fig. 1). In the figure there are four PE's laid out on a 2-D array of gates. Each PE sends a signal through one pixel of a transmissive spatial light modulator (SLM). The SLM is electrically addressed, so that the instructions can come from an

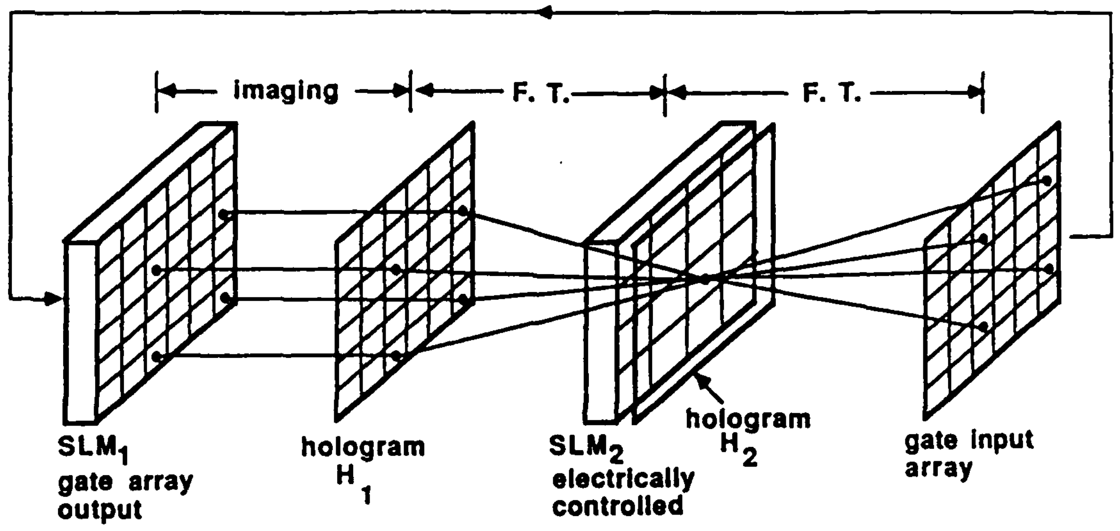


Fig. 7. An optical architecture for the incorporation of superimposed gates for instruction or control bits. The optics are omitted for clarity but are identical to those of Fig. 1. Signals from four gates are shown that fan in to a common control bit.

electronic host. After passing through a common superimposed gate corresponding to the control bit, the signals proceed to the appropriate gate inputs in the gate input array. In this case the second hologram H_2 deflects the signals to the desired gate inputs (gates different from which they came). This optical system is identical to that of Fig. 1 except for the introduction of the SLM for control bits; thus the systems are compatable. Note also that the fanout of each gate in this process is one; a conventional implementation with a large number of PE's would require very high fanout capability or else a tree of gates for each control bit to provide the fanout.

These superimposed gates are not true 3-terminal devices. The common (b) input is regenerated, but the a_i inputs are not. As a result, a design constraint, that these a_i signals do not go through too many superimposed gates in succession without being regenerated by a conventional gate, must be adhered to. This is typically not an issue in the case of control bits. Another consequence is that the total switching energy required for a given processing operation is reduced, because N gates are replaced with one superimposed gate. This is important because it is likely that the total switching energy will ultimately be the limiting factor on the switching speed and number of gates in an optical computer. Other advantages include an increase in computing speed since some of the gates are effectively passive and reduced requirements on the device used to implement the optical gates.

CONCLUSIONS

We have shown that the property of superposition can be exploited in the design of optical or hybrid optical/electronic computing architectures. It can reduce the hologram complexity for highly parallel interconnections, reduce the number of gates in a SIMD system, and permit simultaneous memory access in a parallel shared memory machine, thereby reducing contention problems. Our fundamental reason for studying this is that architectures for optical computing must be designed for the capabilities and limitations of optics; they must not be constrained by the limitations of electronic systems, which have necessarily dominated approaches to digital parallel computing architectures to date.

REFERENCES

Aho, A.V., J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, Mass. Addison-Wesley, 1974.

Borodin, A. and J.E. Hopcroft, "Routing, Merging, and Sorting on Parallel Models of Computation." *Journal of Computer and System Sciences*, Vol. 30, pp. 130-145 1985.

Fortune, S., and J. Wyllie, "Parallelism in Random Access Machines," *Proc 10th Annual ACM STOC*, San Diego, California, pp. 114-118, 1978.

Giles, C. L., and Jenkins, B. K., "Complexity Implications of Optical Parallel Computing," *Proc. Twentieth Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, Calif., to appear, Nov. 1986.

Jenkins, B.K., et al., "Architectural Implications of a Digital Optical Processor," *Applied Optics*, Vol. 23, No. 19, pp. 3465-3474, 1984.

Lev, G., N. Pippenger, and L.G. Valiant, "A Fast Parallel Algorithm for Routing in Permutation Networks" *IEEE Trans. on Computers*, Vol. C-30, No. 2, pp. 33-100, Feb 1981.

Sawchuk, A. A., B.K. Jenkins, C.S. Raghavendra, and A. Varma, "Optical Matrix-Vector Implementations of Crossbar Interconnection Networks," *Proc. International Conference on Parallel Processing*, St. Charles, IL, August 1986; also Sawchuk, et al., "Optical Interconnection Networks," *Proc. International Conference on Parallel Processing*, pp. 388-392, August 1985.

Schwartz, J.T., "Ultracomputers," *A.C.M. Trans on Prog. Lang. and Sys.*, Vol. 2, No. 4, pp. 484-521, October 1980.

Shiloach, Y., and Vishkin, U., "Finding the Maximum, Merging and Sorting in a Parallel Computation Model," *J. Algorithms*, pp. 88-102, March 1981.

Proc. Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, Nov. 1988.

Stochastic Learning Networks for Texture Segmentation*

B.S.Manjunath and R.Chellappa

Signal and Image Processing Institute
Department of EE-Systems
University of Southern California
Los Angeles, CA 90089-0272.

Abstract

In this paper we describe Neural Network based algorithms for the segmentation of textured gray level images. We formulate the problem as one of minimizing an energy function, derived through the representation of textures as Markov Random Fields (MRF). The texture intensity array is modelled as a Gauss Markov Random Field (GMRF) and an Ising model is used to characterize the label distribution. The resulting non-convex energy function is minimized using a Hopfield neural network. The solution obtained is a local optimum in general and may not be satisfactory in many cases. Although stochastic algorithms like simulated annealing have a potential of finding the global optimum, they are computationally expensive. We suggest an alternate approach based on the theory of learning automata which introduces stochastic learning into the iterations of the Hopfield network. A probability distribution over the possible label configurations is defined and the probabilities are updated depending on the final stable states reached by the neural network. This iterated hill climbing algorithm combines the fast convergence of the deterministic relaxation with the sustained exploration of the stochastic algorithms. The performance of this rule in classifying some real textured images is given.

1 Introduction

Neural networks are receiving increasing attention for solving computationally hard optimization problems in computer vision. Their inherent parallelism provides an interesting architecture for implementing many of these algorithms. Few examples include image restoration [1], stereopsis [2] and computing optical flow [3]. The standard Hopfield type networks are designed to minimize certain energy functions and it can be shown that [4] for networks having symmetric interconnections, the equilibrium states correspond to the local minima of the energy function. For practical purposes, networks with few interconnections are preferred because of the large number of processing units required in any image processing application. In this context Markov Random Field (MRF) models for images play an useful role. They are typically characterized by local dependencies and symmetric interconnections which can be expressed in terms of energy functions using Gibbs-Markov equivalence [5].

Texture Segmentation and classification is an important problem in computer vision. Most of the real world objects consist of textured surfaces. One can segment images based on textures even if there are no apparent intensity edges between the different segments. Depending on the nature of the statistics obtained from the image data different segmentation methods are possible. In this paper we use prior models for the conditional intensity distribution and the texture class distribution to segment and classify images consisting of different textures. The conditional distribution of the pixel intensities given the labels is modelled as a fourth order Gauss Markov Random Field (GMRF). The distribution of the labels in the image is characterized by an Ising model. The segmentation can then be formulated as an optimization problem involving minimization of a Gibbs energy function. Finding an optimal solution to this problem requires an exhaustive search over possible label configurations which is practically impossible. It is well known that stochastic relaxation algorithms like simulated annealing [5] can find the global optimum if proper cooling schedules are followed and [6] describes some segmentation algorithms based on this approach. Deterministic relaxation schemes provide very fast solutions but most of the time they get trapped in the local minima. We first describe a neural network algorithm for carrying out the minimization process based on deterministic relaxation. It is observed that models based on GMRF can be easily mapped on to neural networks. The solutions obtained using this method are sensitive to the initial configuration and in many cases starting with a Maximum likelihood estimate is preferred. Stochastic learning can be easily introduced in to the above network and the overall system improves the performance by learning while searching. The learning algorithms used are derived from the theory of stochastic learning automata and we believe that this is the first time such a hybrid system has been used in an optimization problem. The stochastic nature of the system helps in preventing the algorithm from being trapped in a local minimum and we observe that this improves the quality of the solutions obtained.

The organization of this report is as follows : In section 2 the image model is discussed . Section 3 describes the Hopfield Neural network and section 4 provides a brief review of the stochastic learning automata and its application to texture segmentation. Experimental results are given in section 5.

*Partially supported by the AFSOR grant no. 86-0196.

2 Image Model

We use a fourth order GMRF to represent the conditional probability density of the image intensity array given its texture labels. The texture labels are assumed to obey a first or second order Ising Model with a single parameter β, which measures the amount of cluster between adjacent pixels.

Let Ω denote the set of grid points in the $M \times M$ lattice, i. e., $\Omega = \{(i,j) \, , \, 1 \le i,j \le M\}$. Following Geman and Graffigne [7] we construct a composite model which accounts for texture labels and gray levels. Let $\{L_s \, , \, s \in \Omega\}$ and $\{Y_s \, , \, s \in \Omega\}$ denote the labels and zero mean gray level arrays respectively. The zero mean array is obtained by subtracting the local mean computed from a small window centered at each pixel. Let N_s denote the symmetric fourth order neighborhood of a site s. Then we can write the following expression for the conditional density of the intensity at the pixel site s:

$$P(Y_s = y_s | Y_r = y_r, r \in N_s, L_s = l) = \frac{e^{-U(Y_s = y_s | Y_r = y_r, L_s = l)}}{Z(l)}$$

where $Z(l)$ is the partition function of the conditional Gibbs distribution, $r \in N_s$ and

$$U(Y_s = y_s \mid Y_r = y_r, r \in N_s, L_s = l) = \frac{1}{2\sigma_l^2}(y_s^2 - 2 \sum_{r \in N_s} \Theta^l_{s-r} y_s y_r) \qquad (1)$$

In (1), σ_l and Θ^l are the GMRF model parameters of the l-th texture class. The model parameters satisfy $\Theta^l_{r,s} = \Theta^l_{r-s} = \Theta^l_{s-r} = \Theta^l_r$.

The Gibbs energy function computed in (1) should be used in the classification process. However seldom do the texture features tend to be so small as to be captured by a fourth order neighborhood. Increasing the order of the GMRF model requires the estimation of additional model parameters which are quite sensitive. An alternative approach is to calculate the joint distribution of the intensity conditioned on the texture label in a small window centered at the pixel site. The corresponding Gibbs energy can then be used in the relaxation process for segmentation. We view the image intensity array as composed of a set of overlapping $k \times k$ windows W_s, centered at each pixel $s \in \Omega$. In each of these windows we assume that the texture label L_s is homogeneous (all the pixels in the window belong to the same texture). As before we model the intensity in the window by a fourth order stationary GMRF. The local mean is computed by taking the average of the intensities in the window W_s and is subtracted from the original image to get the zero mean image. All our references to the intensity array corresponds to the zero mean image. Let $\mathbf{Y}_s^*$ denote the 2-D vector representing the intensity array in the window W_s. Using the Gibbs formulation and assuming a free boundary model, the joint probability density in the window W_s can be written as,

$$P(\mathbf{Y}_s^* | L_s = l) = \frac{e^{-U_1(\mathbf{Y}_s^* | L_s = l)}}{Z_1(l)}$$

where $Z_1(l)$ is the partition function and

$$U_1(\mathbf{Y}_s^* | L_s = l) = \frac{1}{2k^2\sigma_l^2} \sum_{r \in W_s} \left\{ y_r^2 - 2 \sum_{\tau \in N^* | r+\tau \in W_s} \Theta^l_\tau y_r y_{r+\tau} \right\} \qquad (2)$$

N^* is the set of shift vectors corresponding to a fourth order neighborhood system:

$$\begin{aligned} N^* &= \{\tau_1, \tau_2, \tau_3, \cdots, \tau_{10}\} \\ &= \{(0,1),(1,0),(1,1),(-1,1),(0,2),(2,0),(1,2),(2,1), \\ &\quad (-1,2),(-2,1)\} \end{aligned}$$

The label array is modelled as a first or second order Ising distribution. If $\hat{N}_s$ denotes the appropriate neighborhood for the Ising model, then we can write the distribution function for the texture label at site s conditioned on the labels of the neighboring sites as:

$$P(L_s | L_r \, , \, r \in \hat{N}_s) = \frac{e^{-U_2(L_s \mid L_r)}}{Z_2}$$

where Z_2 is a normalizing constant and

$$U_2(L_s \mid L_r, \, r \in \hat{N}_s) = -\beta \sum_{r \in \hat{N}_s} \delta(L_s - L_r) \, , \, \beta > 0 \qquad (3)$$

In (3), β determines the degree of clustering, and $\delta(i-j)$ is the Kronecker delta. Using the Bayes rule, we can write

$$P(L_s | \mathbf{Y}_s^*, L_r, r \in \hat{N}_s) = \frac{P(\mathbf{Y}_s^* | L_s) P(L_s | L_r)}{P(\mathbf{Y}_s^*)} \qquad (4)$$

Since $\mathbf{Y}_s^*$ is known, the denominator in (4) is just a normalizing factor. The numerator is a product of two exponential functions and can be expressed as,

$$P(L_s \mid \mathbf{Y}_s^*, \, L_r, \, r \in \hat{N}_s) = \frac{1}{Z_p} e^{-U_p(L_s \mid \mathbf{Y}_s^*, \, L_r)} \qquad (5)$$

where Z_p is a normalizing factor and $U_p(.)$ is the posterior energy corresponding to (5). From (1) and (2) we can write

$$U_p(L_s | \mathbf{Y}_s^*, L_r, r \in \hat{N}_s) = w(L_s) + U_1(\mathbf{Y}_s^* | L_s) + U_2(L_s | L_r) \quad (6)$$

Note that the second term in (6) relates the observed pixel intensities to the texture labels and the last term specifies the label distribution. The bias term $w(L_s) = \log Z_1(L_s)$ is dependent on the texture class and it can be explicitly evaluated for the GMRF model considered here using the toroidal boundary assumption. However the computations become very cumbersome if toroidal assumptions are not made. An alternate approach is to estimate the bias from the histogram of the data as suggested by Geman and Graffigne [7]. Finally, the posterior distribution of the texture labels for the entire image given the intensity array is

$$P(\mathbf{L} \mid \mathbf{Y}^*) = \frac{P(\mathbf{Y}^* \mid \mathbf{L}) \, P(\mathbf{L})}{P(\mathbf{Y}^*)} \qquad (7)$$

Maximizing (7) gives the optimal Bayesian estimate. Though it is possible in principle to compute the righthand side of (7) and find the global optimum, the computational burden involved is so enormous that it is practically impossible to do so. However we note that the stochastic relaxation algorithms like simulated annealing require only the computation of (5) to obtain the optimal solution. The network relaxation algorithm in section 3 also uses these values, but is guaranteed to find only the local minima.

3 A Neural Network for Texture Classification

In this section we consider a deterministic relaxation algorithm based on the image model described above. This algorithm can be implemented in a highly parallel fashion on a neural network architecture and typically converges in 20-30 iterations.

We begin by describing a network for the segmentation problem and the energy function it minimizes. This energy function is obtained from the image model described in (2). For convenience of notation let $U_1(i,j,l) = U_1(Y_s^*, L_s = l) + w(l)$ where $s = (i,j)$ denotes a pixel site and $U_1(\, . \,)$ and $w(l)$ are as defined in (6). The network consists of K layers, each layer arranged as an $M \times M$ array, where K is the number of texture classes in the image and M is the dimension of the image. The elements (neurons) in the network are assumed to be binary and are indexed by (i,j,l) where $(i,j) = s$ refers to their position in the image and l refers to the layer. The (i,j,l)-th neuron is said to be ON if its output V_{ijl} is 1, indicating that the corresponding site $s = (i,j)$ in the image has the texture label l. Let $T_{ijl;i'j'l'}$ be the connection strength between the neurons (i,j,l) and (i',j',l') and I_{ijl} be the input bias current. Then a general form for the energy of the network is [4]

$$E = -\frac{1}{2}\sum_{i,j=1}^{M}\sum_{l=1}^{K}\sum_{i'j'=1}^{M}\sum_{l'=1}^{K} T_{ijl;i'j'l'}V_{ijl}V_{i'j'l'} - \frac{1}{2}\sum_{i,j=1}^{M}\sum_{l=1}^{K} I_{ijl}V_{ijl} \tag{8}$$

From our discussion in section 2 we note that an approximate solution for the MAP estimate can be obtained by minimizing (6) for each site in the image. It is easy to see that this is equivalent to minimizing the following energy function for the network:

$$E = \frac{1}{2}\sum_{s=(i,j)}\sum_{l=1}^{K} U_1(i,j,l)V_{ijl} - \frac{\beta}{2}\sum_{l=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{(i',j')\in\hat{N}_{ij}} V_{i'j'l}V_{ijl} \tag{9}$$

where $\hat{N}_{ij}$ is the neighborhood of site (i,j) (same as the $\hat{N}_s$ in section 2). In (9), it is implicitly assumed that each pixel site has a unique label, i.e. only one neuron is active in each column of the network. This constraint can be implemented in different ways. A simple method is to use a *winner-takes-all* circuit for each column so that the neuron receiving the maximum input is turned on and the others are turned off. Alternately a penalty term can be introduced in (9) to represent the constraint as in [4]. From (8) and (9) we can identify the parameters for the network,

$$T_{ijl;i'j'l'} = \begin{cases} \beta & \text{if } (i',j') \in \hat{N}_{ij}, \forall\, l \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

and the bias current

$$I_{ijl} = -\,U_1(i,j,l)$$

The input-output relation can be stated as follows: Let u_{ijl} be the potential of neuron (i,j,l). (Note:l is the layer number corresponding to texture class l) , then

$$u_{ijl} = \sum_{i'=1}^{M}\sum_{j'=1}^{M}\sum_{l'=1}^{K} T_{ijl;i'j'l'}\, V_{i'j'l'} + I_{ijl} \tag{11}$$

and

$$V_{ijl} = \begin{cases} 1 & \text{if } u_{ijl} = \min_{l'}\{u_{ijl'}\} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

In (10) we have no self feedback ,i.e. $T_{ijl;ijl} = 0, \forall i,j,l$ and all the connections have equal strengths. The updating scheme ensures that at each stage the energy decreases. Since the energy is bounded, the convergence of the above system is assured but the stable state will in general be a local optimum.

This neural model is one version of the Iterated Conditional Mode algorithm (ICM) of Besag [8]. This algorithm maximizes the conditional probability $p(L_s = l | Y_s^*, L_{s'}, s' \in \hat{N}_s)$ during each iteration . ICM is a local deterministic relaxation algorithm and very easy to implement. We observe that in general many algorithms based on MRF models can be easily mapped on to Neural networks with local interconnections.

4 Stochastic Learning Algorithms

We begin with a brief introduction to the Stochastic Learning Automaton (SLA). A SLA is a decision maker operating in a random environment. A stochastic automaton can be defined by a quadruple (α, Q, T, R) where $\alpha = \{\alpha_1, \ldots, \alpha_N\}$ is the set of available actions to the automaton. The action selected at time t is denoted by $\alpha(t)$. $Q(t)$ is the state of the automaton at time t and consists of the action probability vector $p(t) = [p_1(t), \ldots, p_N(t)]$ where $p_i(t) =$ prob $(\alpha(t) = \alpha_i)$ and $\sum_i p_i(t) = 1\ \forall t$. The environment responds to the action $\alpha(t)$ with a $\lambda(t) \in R$, R being the set of environment's responses. The state transitions of the automaton are governed by the learning algorithm T, $Q(t+1) = T(Q(t), \alpha(t), \lambda(t))$. Without loss of generality it can be assumed that $R = [0,1]$, i.e., the responses are normalized to lie in the interval [0,1], '1' indicating a complete *success* and '0' total *failure*. The goal of the automaton is to converge to the optimal action, i.e. the action which results in the maximum expected reward. Again without loss of generality let α_1 be the optimal action and $d_1 = E[\lambda(t) \mid \alpha_1] = \max_i\{E[\lambda(t) \mid \alpha_i]\}$. At present no learning algorithms exist which is optimal in the above sense. However we can choose the parameters of certain learning algorithms so as to realize a response as close to the optimum as desired. This condition is called ϵ-optimality. If $M(t) \triangleq E[\lambda(t) \mid p(t)]$, then a learning algorithm is said to be ϵ-optimal if it results in a $M(t)$ such that

$$\lim_{t\to\infty} E[M(t)] > d_1 - \epsilon \tag{13}$$

for a suitable choice of parameters and for any $\epsilon > 0$. One of the simplest learning schemes is the Linear Reward-Inaction rule , L_{R-I} . Suppose at time t we have $\alpha(t) = \alpha_i$ and if $\lambda(t)$ is the response received then according to the L_{R-I} rule,

$$\begin{aligned} p_i(t+1) &= p_i(t) + a\,\lambda(t)\,[1 - p_i(t)] \\ p_j(t+1) &= p_j(t)[1 - a\,\lambda(t)\,p_j(t)] \\ &\quad \forall j \neq i \end{aligned} \tag{14}$$

where a is a parameter of the algorithm controlling the learning rate. Typical values for a are in the range 0.01-0.1. It can be shown that this L_{R-I} rule is ϵ - *optimal* in all stationary environments i.e., there exists a value for the parameter a so that condition (13) is satisfied.

Collective behavior of a group of automata has also been studied. Consider a team of N automata $A_i (i = 1,..,N)$ each having r_i actions $\alpha^i = \{\alpha_1^i \dots \alpha_{r_i}^i\}$. At any instant t each member of the team makes a decision $\alpha^i(t)$. The environment responds to this by sending reinforcement signal $\lambda(t)$ to all the automata in the group. This situation represents a cooperative game among a team of automata with identical payoff. All the automata update their action probability vectors according to (3) using the same learning rate and the process repeats . Local convergence results can be obtained in case of stationary random environments. Variations of this rule have been applied to complex problems like decentralized control of Markov Chains [9] and relaxation labelling [10]

The texture classification discussed in the previous sections can be treated as a relaxation labelling problem and stochastic automata can be used to learn the labels (texture class) for the pixels. A Learning Automaton is assigned to each of the pixel sites in the image. The actions of the automata correspond to selecting a label for the pixel site to which it is assigned. Thus each automaton has K actions and a probability distribution over this action set. Initially the labels are assigned randomly with equal probability. Since the number of automata involved is very large, it is not practicable to update the action probability vector at each iteration. Instead we combine the iterations of the neural network described in the previous section with the stochastic learning algorithm. This results in an iterative hill climbing type algorithm which combines the fast convergence of deterministic relaxation with the sustained exploration of the stochastic algorithm. The stochastic part prevents the algorithm from getting stuck in a local minima and at the same time "learns" from the search by updating the state probabilities. However unlike simulated annealing, we cannot guarantee convergence to the global optimum. Each cycle now has two phases: The first phase consists of the deterministic relaxation network converging to a solution. The second phase consists of the learning network updating its state ,the new state being determined by the equilibrium state of the relaxation network. A new initial state is generated by the learning network depending on its current state and the cycle repeats. Thus relaxation and learning alternate with each other. After each iteration the probability of the more stable states increases and because of the stochastic nature of the algorithm the possibility of getting trapped in a bad local minima is reduced. The algorithm is summarized below :

4.1 Learning Algorithm

Let the pixel site be denoted (as in section 2) by $s \in \Omega$ and the number of texture classes be L. Let A_s be the automaton assigned to site s and the action probability vector of A_s be $\mathbf{p}_s(t) = [p_{s,1}(t), \dots, p_{s,L}(t)]$ and $\sum_l p_{s,l}(t) = 1 \forall s,t$, where $p_{s,l}(t) = \text{prob(label of site } s = l)$. The steps in the algorithm are:

1. Initialize the action probability vectors of all the automata:

$$p_{s,l}(0) = 1/K. \ \forall s, l$$

Initialize the iteration counter to 0.

2. Choose an initial label configuration sampled from the distribution of these probability vectors.

3. Start the neural network of section 3 with this configuration.

4. Let l_s denote the label for site s at equilibrium. Let the current time (iteration number) be t. Then the action probabilities are updated as follows:

$$\begin{aligned} p_{s,l_s}(t+1) &= p_{s,l_s}(t) + a\ \lambda(t)\ [1 - p_{s,l_s}(t)] \\ p_{s,j}(t+1) &= p_{s,j}(t)[1 - a\ \lambda(t) p_j(t)] \\ & \qquad \forall s \text{ and } \forall j \neq l_s \end{aligned} \tag{15}$$

The response $\lambda(t)$ is derived as follows: Suppose the present label configuration resulted in a lower energy state compared to the previous one then it results in a $\lambda(t) = \lambda_1$ and if the energy increases we have $\lambda(t) = \lambda_2$ with $\lambda_1 > \lambda_2$. In our simulations we have used $\lambda_1 = 1$ and $\lambda_2 = 0.25$.

5. Generate a new configuration from this updated label probabilities, increment the iteration counter and goto step 3.

Thus the system consists of two layers, one for relaxation and the other for learning. The relaxation network is similar to the one considered in section 3, the only difference is that the initial state is decided by the learning network. The learning network consists of a team of automata and learning takes place at a much lower speed than relaxation with fewer number of updating. The probabilities of the labels corresponding to the final state of the relaxation network are increased according to (15). Using these new probabilities a new configuration is generated. Since the response does not depend on time, this corresponds to a stationary environment and as we have noted before this L_{R-I} algorithm can be shown to converge to a stationary point, not necessarily the global optimum.

5 Experimental Results

The algorithms are tested on real textures consisting of wood, wool, calf skin, pig skin, sand and grass . Previously computed texture parameters are used in the experiment. The energy functions are obtained by constructing a 11×11 window around each of the pixel sites and computing the Gibb's measure corresponding to the joint distribution in the window. It is assumed that the texture is homogeneous within the window. The bias values for the various textures $w(l_k)$ are chosen by trial and error. These values depend on the different textures present in the image and a discussion regarding their estimation can be found in [6]. The resulting segmentation is sensitive to the bias weights, particularly for the pigskin and sand textures as their properties are very similar. Larger values of β in (3) favours more homogeneous patches in the segmented image and we used values ranging between 0.3 and 2.0 in our experiments . The algorithms are tested on three images. The first two are two class problems consisting of calf skin and grass textures. The resulting classification is shown in figure (1). The results for the six class problem are shown in figure (2).

The deterministic relaxation scheme of section 3 usually takes about 20-40 iterations to converge. For comparison purposes, the percentage misclassification for example 2 was computed for the various algorithms. The deterministic relaxation resulted in an error of about 15% compared to 22% for the Maximum likelihood method. For the learning algorithm the error was 8.7%. Compared to this the simulated annealing algorithm [6] had a misclassification of about 6.8%. Simulated annealing is atleast 10-15 times computionally more expensive than the deterministic algorithms. It was also observed that the deterministic relaxation algorithm performs better when started with a random configuration than with maximum likelihood estimates. In terms of the classification error for example 2, this difference was about 1%. Also when all the neurons in the deterministic relaxation scheme were updated simultaneously, the system used to converge to limit cycles ,switching between two nearby states. Such cycles must be identified when considering parallel implementation of neural networks on computers. This will not be a problem in case of the analog networks as the probability of such an event happening in a physical system is zero.

6 Conclusions

In this paper we have described learning and neural network algorithms for texture classification. The deterministic relaxation (ICM) algorithm can be easily mapped to a neural network for minimizing the energy function and reasonably good solutions can be obtained quickly. Texture classification is treated as a relaxation labelling problem and a learning system is developed to learn the pixel classes. Convergence of both these schemes to local optima can be proved . It is observed that learning is a very slow process and hence cannot be directly used in the segmentation process. A combination of learning and deterministic relaxation seems to improve the quality of solutions obtained and performs well compared to simulated annealing in terms of speed.

The learning algorithm described in this paper is very general and can be applied in a variety of situations. This model might be particularly useful if little is known about the image models and a good criterion function is available for generating a reinforcement signal. Currently we are working on extending this method to hierarchical segmentation and other image processing applications.

Acknowledgements: We wish to thank Tal Simchony for many useful discussions and in helping to formulate the image model.

References

[1] Y.T. Zhou, R. Chellappa,A. Vaid and B.K. Jenkins, "Image Restoration Using a Neural Network", *IEEE Trans. Accous.. Speech and Signal Processing*, vol. ASSP-36(7), pp. 1141-1151, July 1988 1988.

[2] Y.T. Zhou and R. Chellappa, "Stereo Matching Using a Neural Network ", In *Proc. IEEE International Conference on Accoustics, Speech and Signal Processing*, New York,NY, April 1988.

[3] Y.T. Zhou and R. Chellappa, "Computation of Optical Flow Using a Neural Network ", In *Proc. IEEE International Conference on Neural Networks*, SanDiego, California, July 1988.

[4] J.J. Hopfield and D.W. Tank, "Neural computation of decision in optimization problems", *Biological Cybernetics*, vol.52, pp. 114-152, 1985.

[5] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images", *IEEE Transactions on Pattern Analysis and Machine Inteligence*, vol.6, pp. 721-741, November 1984.

[6] T. Simchony and R. Chellappa, "Stochastic and deterministic algorithm for texture segmentation", In *Proc. IEEE International Conference on Accoustics, Speech and Signal Processing*, New York,NY, April 1988.

[7] S. Geman and C. Graffigne, "Markov Random Fields Image Models and their Application to Computer Vision", In *Proc. of the International Congress of Mathematicians 1986*, Ed. A.M. Gleason .American Mathematical Society ,Providence, 1987.

[8] J. Besag, "On the statistical analysis of dirty pictures", *Journal of Royal Statistic Society B*, vol. 48 No. 6, pp. 259-302, 1986.

[9] Wheeler and K.S. Narendra, "Decentralized Learning in Finite Markov Chains", *IEEE Trans. Automatic Control*, vol. AC-31(6), pp. 519-526, June 1986.

[10] M.A.L. Thathachar and P.S. Sastry, "Relaxation Labelling with Learning Automata ", *IEEE Trans. Pattern analysis and Machine Intelligence*, vol. PAMI-8(2), pp. 256-268, March 1986.

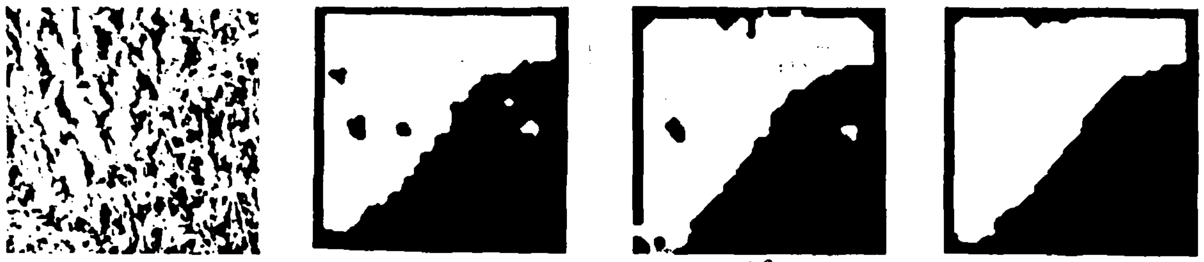

1.1 1.2 1.3 1.4

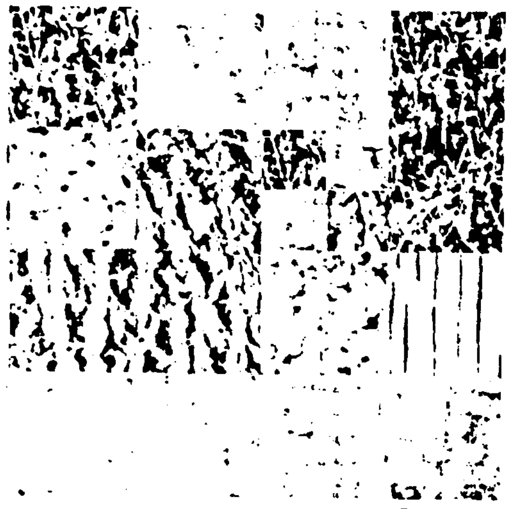
Figure 1: Two class segmentation.

1.1 Original Image.

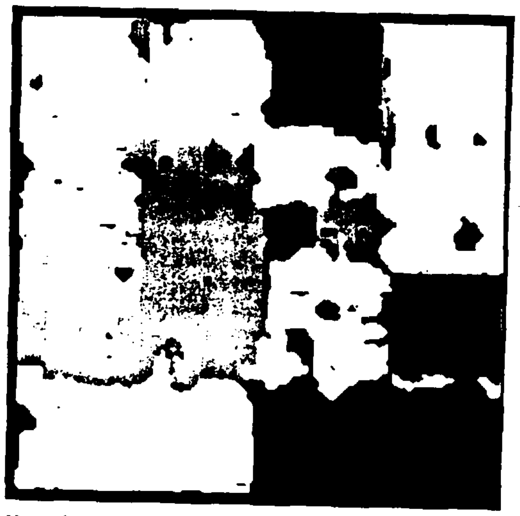
1.2 Neural network solution with ML estimate as initial condition.

1.3 Neural network solution with random initial condition.

1.4 Neural network with stochastic learning.


2.1 Original Image.

2.2 Neural network solution with ML estimate as initial condition.


2.3 Neural network solution with random initial condition.

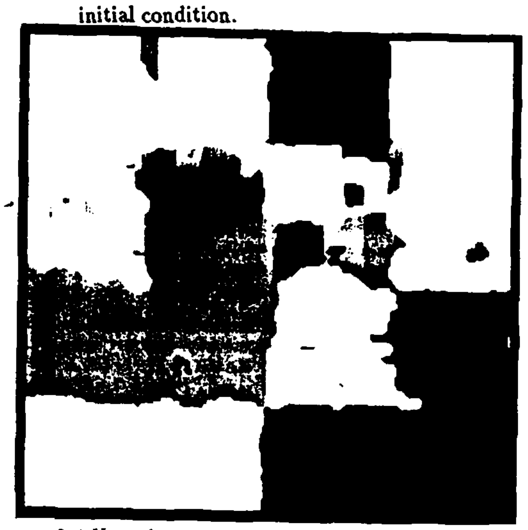

2.4 Neural network with stochastic learning.

Figure 2: Six class segmentation.

Reprinted from Optics Letters, Vol. *13*, page 892, October 1988.

Model for an incoherent optical neuron that subtracts

B. K. Jenkins and C. H. Wang

Signal and Image Processing Institute, Department of Electrical Engineering, University of Southern California, Los Angeles, California 90089-0272

Received January 4, 1988; accepted July 1, 1988

An incoherent optical neuron is proposed that subtracts inhibitory inputs from excitatory inputs optically by utilizing two separate device responses. Functionally it accommodates positive and negative weights, excitatory and inhibitory inputs, and nonnegative neuron outputs, and it can be used in a variety of neural network models. An extension is given to include bipolar neuron outputs in the case of fully connected networks.

In this Letter we propose a general incoherent optical neuron (ION) model that can process excitatory and inhibitory signals optically without electronic subtraction. Conceptually, the inhibitory signal represents a negative signal with a positive synaptic weight or a positive signal with a negative synaptic weight. The ION can be used in a network in which the neuron outputs are nonnegative and the synaptic weights are bipolar, for example, by connecting the interconnections with negative weights to the inhibitory neuron inputs and those with positive weights to the excitatory inputs. Our intent is to show that it is in principle not necessary to go to optoelectronic devices solely because of the requirement for subtraction capability.

Techniques that have been described to date are impractical in all-optical implementations of most neural networks. They utilize an intensity and/or weight bias, in some cases coupled with complemented weights or inputs. As noted in Ref. 1, these techniques suffer from bias buildup and/or thresholds that must vary from neuron to neuron. A technique described in Ref. 2 eliminates most of these drawbacks in the special case of fully connected networks.

The ION model uses separate device responses for inhibitory and excitatory inputs. This is modeled after the biological neuron, which processes the excitatory and inhibitory signals by different mechanisms (e.g., chemical-selected receptors and ion-selected gate channels).[3] The ION comprises two elements: an inhibitory, *I*, element and a nonlinear output, *N*, element. The inhibitory element provides inversion of the sum of the inhibitory signals; the nonlinear element operates on the sum of the excitatory signals, the inhibitory element output, and an optical bias to produce the output of the neuron. The inhibitory element is linear; the nonlinear threshold of the neuron is provided entirely by the nonlinear output element. Figures 1(a) and 1(c) show the characteristic curve of the *I* and *N* elements, respectively. The structure of the ION model is illustrated in Fig. 1(d). The input–output relationships of the *I* and *N* elements are

$$I_{\text{out}}^{(I)} = \hat{I}_{\text{inh}} = 1 - I_{\text{inh}}, \tag{1}$$

$$I_{\text{out}}^{(N)} = \psi[I_{\text{in}}^{(N)} - \alpha] = \psi(\hat{I}_{\text{inh}} + I_{\text{exc}} + I_{\text{bias}} - \alpha), \tag{2}$$

where I_{inh} and I_{exc} represent the total inhibitory and excitatory inputs, respectively, $I_{\text{in}}^{(N)}$ is the total input to the *N* elements, I_{bias} is the bias term for the *N* element, which can be varied to change the threshold, and α is the offset of the characteristic curve of the *N* element. $\psi(\cdot)$ denotes the nonlinear output function of the neuron. If we choose I_{bias} to be $\alpha - 1$, the output of the *N* element is

$$I_{\text{out}}^{(N)} = \psi(I_{\text{exc}} - I_{\text{inh}}), \tag{3}$$

which is the desired subtraction. In general, the *I* element will not be normalized [Fig. 1(b)], in which case the offset, a_1, and the slope of its response can be compensated by setting $I_{\text{bias}} = \alpha - a_1$ and attenuating

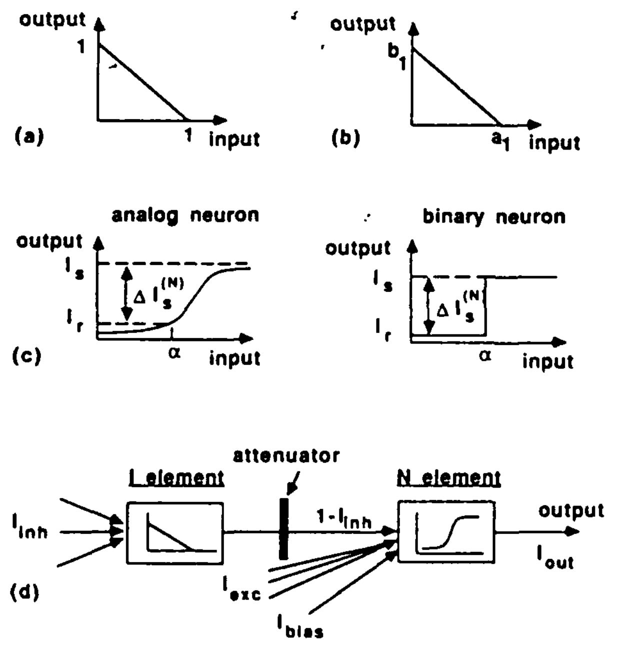


Fig. 1. The ION. (a) The inhibitory element; (b) the unnormalized inhibitory element; (c) the nonlinear element; (d) the structure.

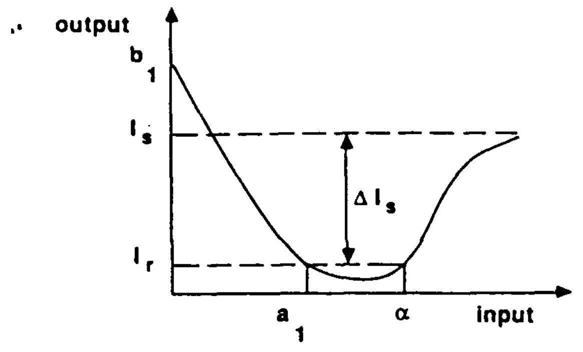


Fig. 2. Characteristic response of a Hughes twisted-nematic liquid-crystal light valve, a possible device for the homogeneous ION model.

the output of the I element by a factor of b_1/a_1, respectively. The unnormalized I element must have gain greater than or equal to 1 A nonzero neuron threshold θ can be implemented by shifting the bias by the same amount, so $I_{\text{bias}} = \alpha - a_1 - \theta$ for the unnormalized I element.

The ION model can be implemented by using separate devices for the I and N elements as depicted in Fig. 1 (the heterogeneous case) or by using a single device with a nonmonotonic response (Fig. 2) to implement both elements (the homogeneous case). Possible devices for ION implementation include bistable optical arrays and spatial light modulators such as liquid-crystal light valves. A single Hughes liquid-crystal light valve could implement both elements. The offset of the device response must satisfy $\alpha \geq na_1 + \theta$, where $n = 1$ for a heterogeneous implementation and $n = 2$ for a homogeneous implementation.

A device to realize the inhibitory I element will, of course, not have a perfectly linear response. To assess the robustness of this model to nonlinear I elements and compensation techniques for large deviations from an ideal response, we have performed simulations of a network similar to Grossberg's competitive network[4] for edge detection. The simulated network contains 30 conventional inner-product neurons connected in a ring structure with input and lateral on-center–off-surround connections. We model the normalized I element response as $\exp[-(x/a)^b]$, where a and b are parameters that determine the specific nonlinear response. This provides insight into the sensitivity of the ION model to nonlinearities in the I element response without being overly specific to one given device. A suitable choice of a and b does provide a close fit to the inversion region of the normalized experimental characteristic of a liquid-crystal light valve. By adjusting the parameters a and b, four different nonlinear inversion curves were simulated in this network. In the simulation a compensating attenuator was used before the I element instead of after it. The N element response, which provides a close approximation to the normalized increasing portion of the liquid-crystal light valve response (Fig. 2), is modeled as $1 - \exp[-(x/0.43)^{1.2}]$. Figure 3 shows the computer-simulated responses of this network. Each resolvable row of Fig. 3 represents a one-dimensional simulation on a distinct one-dimensional input. Thirty different binary inputs were each simulated at four different input signal levels. Figure 3(b) gives the ideal output, and Fig. 3(d) simulates a response that is close to the experimental response of our liquid-crystal light valve. Deviation from linearity of the I element is measured by normalized mean-squared error, which is defined as $\int[v(i) - \hat{v}(i)]^2 \mathrm{d}i / \int v(i)^2 \mathrm{d}i$, where $v(i)$ and $\hat{v}(i)$ are the output values of the linear and simulated nonlinear characteristic curves, respectively. The input level i ranged from 0 to 0.7. Our liquid-crystal light valve characteristic has a normalized mean-squared error of 50%, which does not perform well. If proper input attenuation of the I element is included, the network performs correctly. Four nonlinear curves are simulated, each with optimal input attenuation; we find that deviations from linearity that give a normalized mean-squared error of approximately 15% (measured after input attenuation) can be tolerated. For more extremely nonlinear devices, a bias point and limited region of operation can be used.

The fan in and fan out of the ION, neglecting interconnection effects such as cross talk, can be calculated as follows. (Interconnection effects are important but are not peculiar to the ION model.) We assume binary neurons. As shown in Fig. 1(c), the output of the ith neuron can be formulated as $I_r + \Delta I_s^{(N)} V_i$, where $V_i \in \{0, 1\}$ is the output state of neuron i and I_r is the residual output of element N. Let the fan in and fan out of each neuron be N_{in} and N_{out}, respectively. The summed inputs to neuron j can be grouped into two terms, a noise term caused by residual outputs (I_r) of the optical neurons and the signal term. Consider the worst case, i.e., all weights are close to one and only one input is active. If we assume that each neuron must be able to discriminate a change in any one of its input lines, then the signal term must at least be greater than the noise term. This is a reasonable assumption

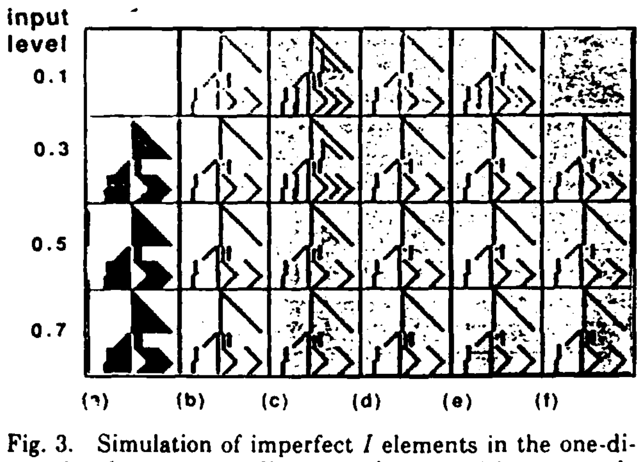


Fig. 3. Simulation of imperfect I elements in the one-dimensional on-center–off-surround competitive network. (S_1 is the input attenuation factor for the I element; mse is the normalized mean-squared error deviation from linearity of the I element response.) (a) The network input. (b)–(f) The network outputs for (b) the linear I element; (c) $a = 0.46$, $b = 2.1$, $S_1 = 1.0$, mse = 19%; (d) $a = 0.25$, $b = 1.2$, $S_1 = 2.5$, mse = 4%; (e) $a = 0.33$, $b = 1.5$, $S_1 = 1.5$, mse = 14%; (f) $a = 0.16$, $b = 0.9$, $S_1 = 5.0$, mse = 7%.

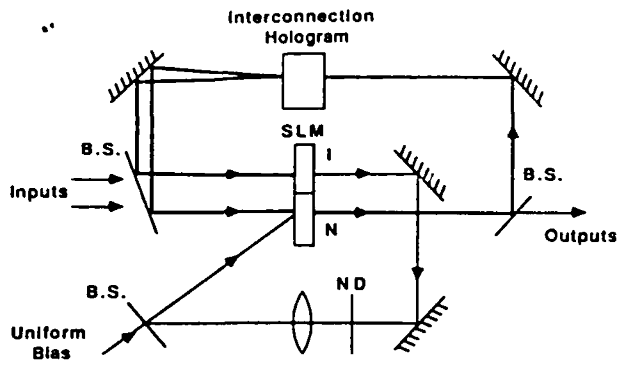


Fig. 4. Single-layer feedback net using a single spatial light modulator, SLM, to implement both I and N elements. B.S., beam splitter; ND, neutral-density filter.

for networks with small fan in and fan out. Thus the maximum fan in is

$$N_{\text{in}}^{(\text{max})} = \frac{\Delta I_s^{(N)}}{I_r} = \text{extinction ratio of element } N. \quad (4)$$

The fan out is calculated from the I element, as shown in Fig. 1(b). The ratio of the maximum input a_1 to the minimum input $I_s^{(N)}/N_{\text{out}}^{(\text{max})}$ is the fan in $N_{\text{in}}^{(\text{max})}$, where $N_{\text{out}}^{(\text{max})}$ is the maximum fan out over all neurons, thus

$$N_{\text{out}}^{\text{max}} = \frac{I_s^{(N)}}{a_1} N_{\text{in}}^{(\text{max})} \approx \frac{\Delta I_s^{(N)}}{a_1} N_{\text{in}}^{(\text{max})}, \quad (5)$$

where the approximation holds when the extinction ratio of the N element is large. For networks with large fan in, we assume instead that the neuron can discriminate a change in a constant fraction β of its input signals. In this case, there are no such limitations on N_{in} and N_{out}. Instead, $1/\beta$ is limited by the extinction ratio of the N element, and the fan out is still related to the fan in of the network by relation (5). For example, in many networks a $1/\beta$ ranging from 10 to 100 may be sufficient for the optical neuron, while the maximum fan in may be 10^3–10^4. During implementation of the ION model, with many optical devices I_s can be varied with the intensity of the read beam, which effectively increases the gain and permits a larger fan out.

As an example, a conceptual diagram of an implementation of a single-layer feedback net is shown in Fig. 4. It utilizes a single two-dimensional spatial light modulator for both I and N elements. The output of the I element is imaged onto the input of the N element after it passes through a neutral-density filter as the (uniform) attenuation. A uniform bias beam is also input to the N element. The N-element output is fed back through an interconnection hologram to the inputs of both I and N elements, representing inhibitory and excitatory lateral connections, respectively.

We now present a variant of the ION model that incorporates bipolar neuron outputs in the case of fully connected networks. The operation of the network is given by

$$\hat{V}_i = \psi\left(\sum_{j=1}^{N} W_{ij} V_j\right), \quad (6)$$

where $V_j \in [-1, 1]$ is the output of the jth neuron at time t, $W_{ij} \in [-1, 1]$ is the normalized weight from neuron j to neuron i, $\hat{V}_i$ is the output of the ith neuron at time $t + 1$, and N is the number of neurons. A special case of this is the bipolar binary neuron used by Amari[5] ($V_j \in \{-1, 1\}$). In this case, the nonlinear output function $\psi(x)$ is equal to 1 for $x \geq 0$, otherwise it is -1.

By using a complementary offset scheme, Eq. (6) can be rewritten as

$$\frac{1}{2}(1 + \hat{V}_i) = \psi\left[\sum_{j=1}^{N} \frac{(1 - W_{ij})}{2} \frac{(1 - V_j)}{2} + \sum_{j=1}^{N} \frac{(1 + W_{ij})}{2} \frac{(1 + V_j)}{2}\right], \quad (7)$$

where $\psi(x)$ is the nonlinear output function of the neuron. All terms in parentheses are positive and can be represented by intensities. The neuron input and output are in the form $(1 + V_i)/2$, and the I element is used to generate the $(1 - V_i)/2$ term. A Hopfield net[6] is identical except for the neuron outputs $V_i \in \{0, 1\}$; for this we can replace $(1 + V_i)/2$ with V_i and $(1 - V_i)/2$ with $\bar{V}_i$ in Eq. (7), where $\bar{V}_i$ is the complement of V_i and is generated by the I element.

Most of this research was presented at the 1987 annual meeting of the Optical Society of America.[7]

References

1. A. F. Gmitro and G. R. Gindi, in *Proceedings of the First International IEEE Conference on Neural Networks* (Institute of Electrical and Electronics Engineers, New York, 1987), p. III-599.
2. R. D. Te Kolste and C. C. Guest, in *Proceedings of the First International IEEE Conference on Neural Networks* (Institute of Electrical and Electronics Engineers, New York, 1987), p. III-625.
3. M. Wang and A. Freeman, *Neural Function* (Little, Brown, Boston, 1987).
4. S. A. Ellias and S. Grossberg, Biol. Cybern. **20**, 69 (1975).
5. S.-I. Amari, IEEE Trans. Comput. **C-21**, 1197 (1972).
6. J. J. Hopfield, Proc. Natl. Acad. Sci. USA **79**, 2554 (1982).
7. B. K. Jenkins and C. H. Wang, J. Opt. Soc. Am. A **4**(13), P127 (1987).

Implementation of a Subtracting Incoherent Optical Neuron

C. H. Wang and B. K. Jenkins
Signal and Image Processing Institute, Department of Electrical Engineering
University of Southern California, Los Angels, CA 90089-0272

Abstract

The **Incoherent Optical Neuron (ION)** model uses two separate incoherent optical device responses to subtract inhibitory inputs from excitatory inputs for general neural networks. The operational considerations of this model, based on a Hughes liquid crystal light valve, are discussed. Experimental demonstration of incoherent subtraction for binary and analog signal levels are presented.

1 Introduction

In this paper we demonstrate the feasibility of a general incoherent optical neuron (ION) model [1] that can process excitatory and inhibitory signals optically without electronic subtraction. An inner-product type optical neuron should perform a nonlinear operation on its weighted sum inputs as shown in Eq. (1). The inputs can be excitatory (positive) or inhibitory (negative).

$$\hat{V}_i = \psi[\sum_{j=1}^{N} W_{ij} V_j] \qquad (1)$$

where $\hat{V}_i$ is the output of neuron i in the same layer, V_j are the signal inputs, W_{ij} are the synaptic weights, and $\psi(\cdot)$ is the output nonlinear function of the neuron, which is a nondecreasing function and has a finite range.

A fully coherent system can subtract signals directly, using differences in phase (or path length) of the optical beams. The tradeoff is that the system must be stable within significantly less than one wavelength ($\sim 0.5\ \mu m$). In addition, the phases of components in the system must be accurately controlled. These factors lead to difficulty in implementing arbitrary neural networks with a fully coherent system.

An incoherent system is more robust in terms of stability, position accuracy requirements, and noise immunity. Existing techniques such as input bias or weight bias methods suffer from an input dependent bias or a threshold that must vary from neuron to neuron [2].

* Proc. IEEE 3rd annual Parallel Processing Symposium, Fullerton, CA., March 1989.

Weight bias:

$$\begin{aligned}\hat{V}_i &= \psi[\sum_{j=1}^{N}(W_{ij}+W_b)V_j] \\ &= \psi[\sum_{j=1}^{N}W_{ij}V_j + W_b\sum_{j=1}^{N}V_j] \qquad (2)\end{aligned}$$

Input bias:

$$\begin{aligned}\hat{V}_i &= \psi[\sum_{j=1}^{N}W_{ij}(V_j+V_b)] \\ &= \psi[\sum_{j=1}^{N}W_{ij}V_j + V_b\sum_{j=1}^{N}W_{ij}] \qquad (3)\end{aligned}$$

In order to eliminate the bias at each pass through a neuron, the threshold of each neuron must depend on its inputs in Eq. (2) and on its weights in Eq. (3). A technique described in [3] eliminates most of these drawbacks in the special case of fully connected networks.

2 The Incoherent Optical Neuron Model

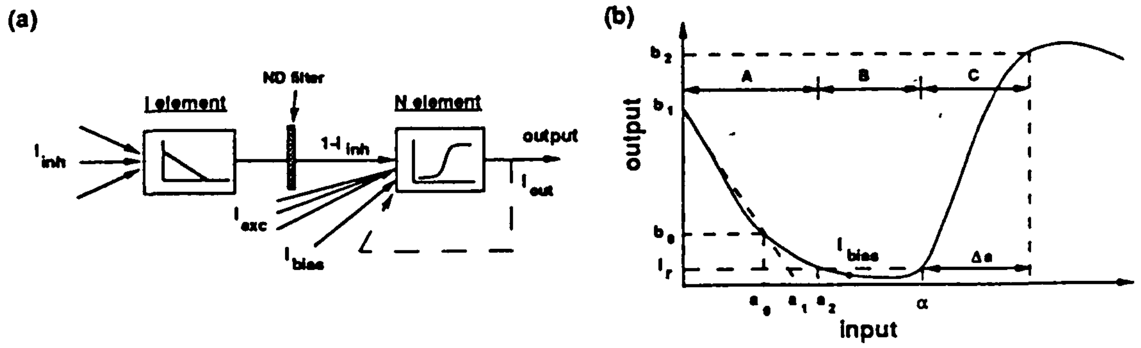


Fig. 1 (a) The ION structure. (b) Typical characteristic of Hughes liquid crystal light valve, serving as both I and N elements. Regions A and C are used for the I and N elements respectively. Region B is used to provide bias for the N element.

The *Incoherent Optical Neuron* (ION) model provides for incoherent subtraction in optical neural networks without the above limitations. It uses separate device responses for inhibitory and excitatory inputs. This is modeled after the biological neuron which processes the excitatory and inhibitory signals by different mechanisms (e.g. chemical-selected receptors and ion-selected gate channels) [5]. The ION comprises two elements: an inhibitory (I) element and a nonlinear output (N) element as shown in Fig. 1(a). The inhibitory element provides inversion of the sum of the inhibitory signals; the nonlinear element operates on the sum of the excitatory signals, the inhibitory element output, and an optical bias to produce the output of the neuron. The inhibitory element is linear; the nonlinear threshold of the neuron is provided entirely by the nonlinear output element.

Figure 2 shows a paradigm for an optical neural network, which uses incoherent optical neurons combined with optical interconnections. A volume hologram can be used to emulate synaptic weights in the optical neural network. The interconnection network should be adaptive during the training phase. Psaltis et. al. [6, 7] have discussed several learning issues in photorefractive crystals. As shown in Fig. 2, the incoherent optical neurons process the weighted sums from the interconnection network; they may also serve as input transducers. The input of the neurons are also fed to the interconnection network to form correlation with the output of the neurons during the learning phase. Then the modified interconnection strength is stored in the interconnection network.

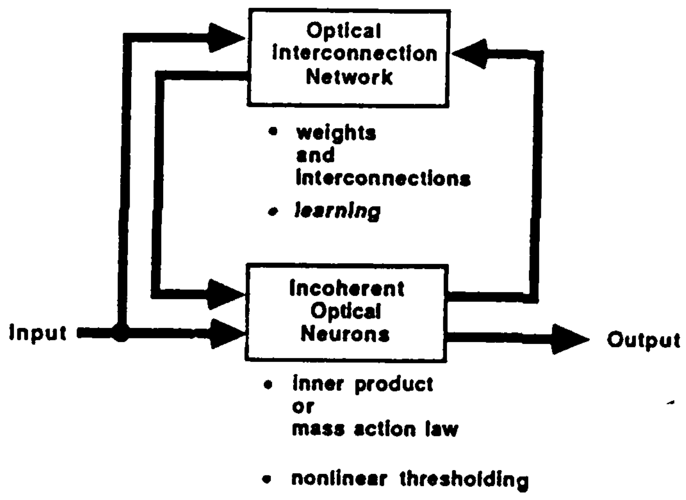

Optical Interconnection Network
• weights and interconnections
• learning
Incoherent Optical Neurons
Input
Output
• inner product or mass action law
• nonlinear thresholding

Fig. 2 A paradigm for an optical neural network.

3 Implementation of the ION model by LCLV

A Hughes liquid crystal light valve (LCLV) is an optical light modulator which can implement approx. 10^5 neurons on one device. A 1.6 x 10^3 gate logic array based on an LCLV was demonstrated by J. Wang and P. Chavel [10]. The typical response time of the LCLV is 30 *ms*. Other devices currently under development, such as ferroelectric LCLVs, have response times on the order of μs [4]. Fig. 1(b) shows a typical characteristic curve of a LCLV, which can be used for both the I (region A) and N (region C) elements. Region B is required to provide bias for the N element. In the remainder of this section we will discuss how an LCLV can be used to implement an array of ION's.

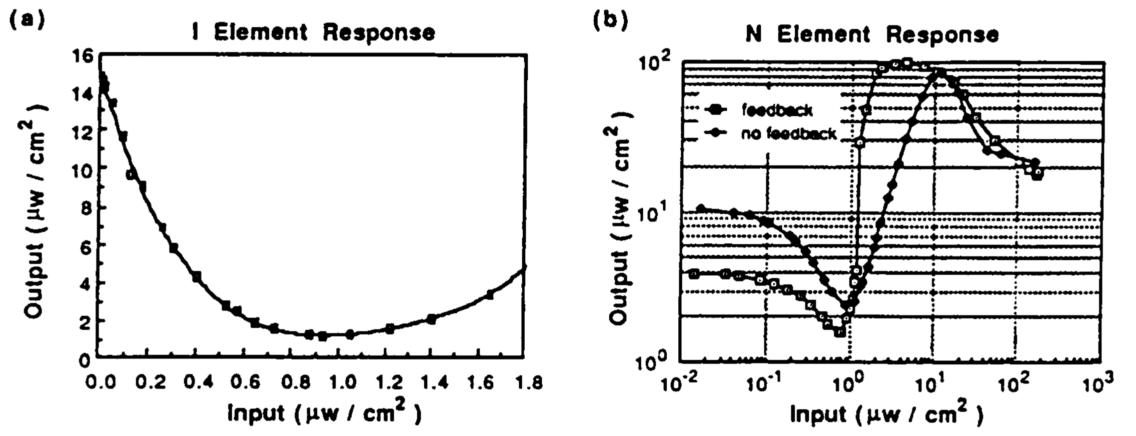


Fig. 3 Characteristics of the (a) I and (b) N elements in the test circuit for the condition V=5.0 volts, f=1.5 Khz, P=200 mw. The I element is fairly linear within 50% of its operation range. In (b), the self-feedback of the N element is necessary to satisfy the ION requirements for this particular device.

Generally, the characteristic of the LCLV in region A is unfortunately not linear (Fig. 3(a)). Let the maximum output of the light valve in region B be I_r , the residue output (Fig 1(b)). Then the maximum operation range of the inhibitory (I) element is between 0 and a_2 [1]. In order to ensure linear subtraction, we need to further limit the operation range of the I element to be $(0, a_0)$. Thus the output of the I element can be modeled as

$$I_{out}^{(I)} = b_1 - \frac{b_1 - a_0}{a_0} I_{inh} \quad . \tag{4}$$

where $I_{inh} \in (0, a_0)$ is the total inhibitory input. For the biasing conditions of our LCLV, f=1.5Khz and V=5.01 volts, if we limit the maximum inhibitory input to be $0.6a_2$, the measured root mean square deviation from linearity is 16%. Simulations we have performed indicate that this amount of nonlinearity in the I element response is acceptable [1]. To match the response of the nonlinear (N) element, the output of the I element is attenuated by a factor γm, where $m = (b_1 - a_0)/a_0$ and is the slope of the I element. If γ is greater than 1, the I element output is attenuated, otherwise we have some gain for the I element. Let the corrected output of the I element be $\hat{I}_{inh} = I_{out}^{(I)}/\gamma m$, then the output of the N element is

$$I_{out}^{(N)} = \psi(I_{in}^{(N)} - \alpha) = \psi(\hat{I}_{inh} + I_{exc} + I_{bias} - \alpha) \tag{5}$$

where $I_{in}^{(N)}$ represents the total input to the N element, I_{exc} is the total excitatory input, I_{bias} is the bias term for the N element, which can be varied to change the threshold, and α is the offset of the characteristic curve of the N element. $\psi(\cdot)$ denotes the nonlinear output function of the neuron. The operation range of the N element is Δa (Fig. 1(b)) which should be equal to the output variation of the I element, a_0/γ, to provide linear subtraction. Since Δa depends on the bias and operating parameters of the light valve, we can adjust γ to fulfill this requirement. The bias of the N element is set to be

$$I_{bias} = \alpha - \frac{b_1}{\gamma m} \tag{6}$$

From Fig 1(b), the bias point, I_{bias}, should be greater than a_2 to prevent from operating the N element in the I element region. Meanwhile, the separation of the operation points of the N and I elements, $\alpha - a_2$, should be greater than a_0/γ; since the output of the I element is nonnegative, i.e. $b_1/\gamma m - a_0/\gamma > 0$, we can instead require

$$\alpha > \frac{b_1}{\gamma m} + a_2. \tag{7}$$

This can be done by adjusting the bias voltage and frequency of the light valve, and by selecting the proper residue output level, I_r. We rewrite the output of the N element, via Eqs. (4), (5), (6), and the definitions of m and $\tilde{I}_{inh}$, as

$$I_{out}^{(N)} = \psi(I_{exc} - \frac{I_{inh}}{\gamma}). \tag{8}$$

If the N element inputs are attenuated by γ, then we get perfect subtraction. Fig. 3(a) and (b) show the characteristic curve of the I and N element respectively. Self-feedback is necessary for the LCLV to fulfill the constraints of the N element response for the ION model.

4 Experimental Results

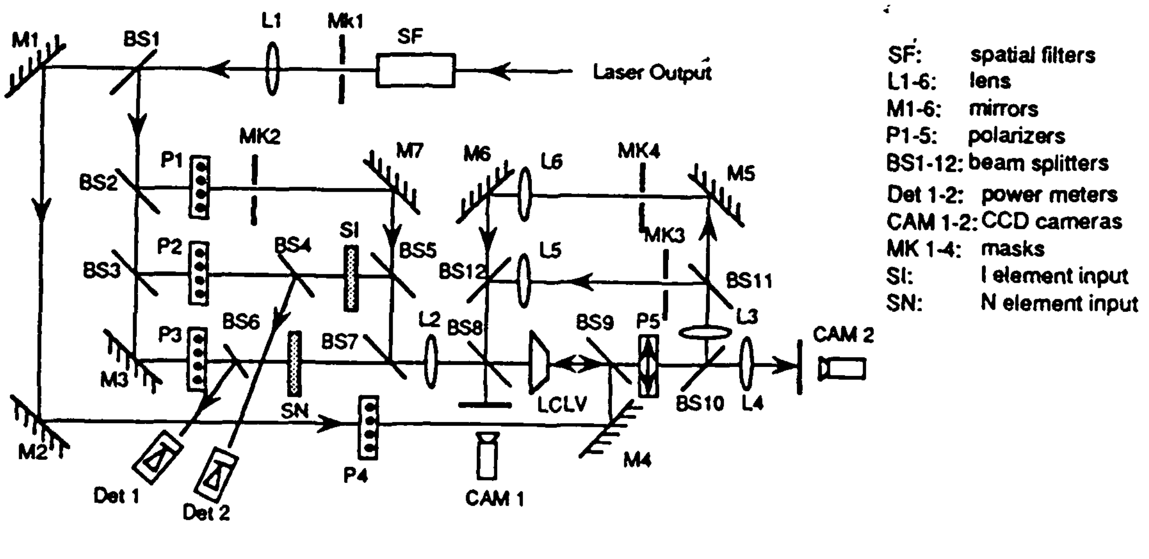
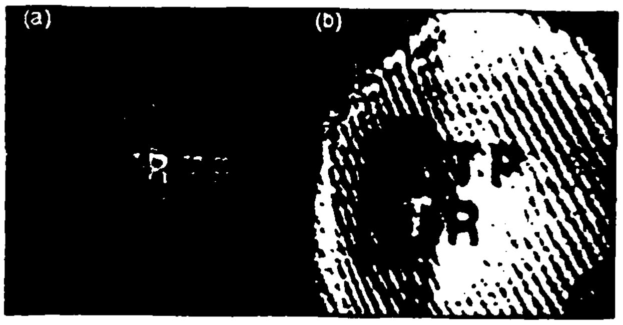


Fig. 4 Experimental setup of the test ION circuit.

Figure 4 shows the experimental setup for implementation and testing of an array of ION's. Three input beams are used to provide N element bias, I element inputs and N element inputs, which are controlled by polarizer pairs P1, P2 and P3 respectively. The I element input path, SI-BS5-BS7-L2-BS8-LCLV, is imaging with a magnification factor of 0.8. The same magnification factor is applied to the N element input path, SN-BS7-L2-BS8-LCLV. Two feedback paths are implemented, one for I to N connection, which is BS9-BS10-L3-BS11-L5-BS12-BS8-LCLV. The other feedback path through mirror M5, M6 is for the N to N self-feedback connection.

Figure 5(a) and (b) shows the experimental result for binary subtraction. For the binary case, two character sets are chosen for the N (left side) and I (right side) inputs (Fig. 5(a)). All four possible cases are included (corresponding to 1 or 0 for the N element input, and 1 or 0 for the I element input). A bias is added to the N element inputs. Figure 5(b) shows I element outputs (right side) and the final neuron outputs (left side; these are also the N element outputs). The ideal result is a portion of the character "R" (right top) and the full character "T" (right bottom) in the N element area, and agrees with Fig. 5(b); these regions correspond to a 1 on the excitatory inputs and a 0 on the inhibitory inputs.



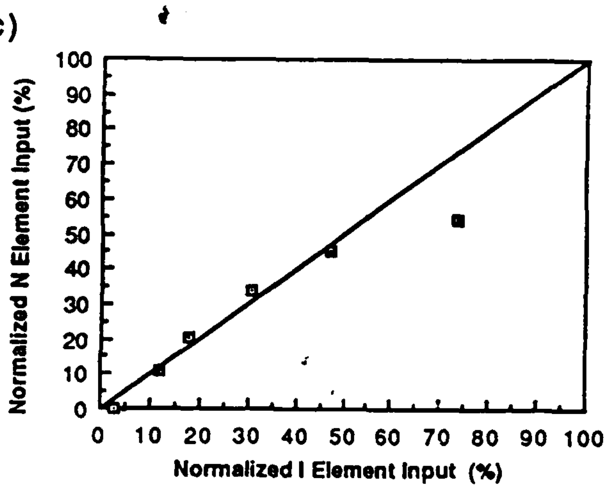


Fig. 5 Results of ION subtraction. Binary subtraction with (a) N input (left) and I input (right) patterns, and (b) the subtraction result. (c) The normalized I input vs. N input for a constant N output, showing grey level response. The subtraction is close to linear if we only use 50 % of the total I input.

To test the subtraction linearity for the grey level case, we set the I input to its minimum and the N input to a small but nonzero value and measure the N element output. For every increment in the I input, we adjust the N input such that we get the same reading for the N element output. Fig 5(c) is a plot of the normalized linear subtraction of the N inputs from I inputs. Essentially, it is equal to the complementary plot of the I element response. Here we use the full operation range of the I element. If we limit the operation of the I inputs to be

50% of a_2, then we obtain subtraction that is very close to linear. In our setup, the range of the attenuated I output is around $0.2\mu w/cm^2$. Since there is significant loss in the feedback ($I \rightarrow N$) path, the nonlinearity of the N element needs to be very steep (i.e. high differential gain) to satisfy the ION requirements; thus the use of self-feedback for the N elements.

5 Discussion and Conclusion

An incoherent subtraction method for neural nets was implemented using a Hughes liquid crystal light valve. Conceptually, the ION model is a general purpose model which can implement a variety of different neuron types. A conceptual example for implementing Hopfield type networks based on the ION model was shown in [1].

The ION model can also implement Grossberg's mass action neuron, which has been used in a variety of neural networks for pattern recognition [8] and visual perception [9]. The mass action neuron can be described as

$$\begin{aligned} \dot{x}_j &= -Ax_j + (B - x_j)I_{exc} - x_j I_{inh} \qquad (9) \\ I_{exc} &= \sum_{i=1}^{n_1} \psi(x_k)C_{ij} + I_j \\ I_{inh} &= \sum_{i=1}^{n_2} \phi(x_k)D_{ij} + J_j \end{aligned}$$

where x_j is the membrane potential of neuron j, and I_{exc} and I_{inh} denote the total excitatory and inhibitory inputs to neuron j. $\psi(x)$ and $\phi(x)$ are the output of current neuron and its inhibitory interneuron respectively, and are sigmoid type functions in most cases. I_j and J_j are the excitatory and inhibitory inputs from other layers. A and B are the decay constant and maximum membrane potential, respectively. C_{ij} and D_{ij} are the interconnection weights. The above equation can be grouped into two terms to be implemented by I and N elements respectively. For the discrete case, we can rewrite it as

$$x_j(k+1) = x_j(k)[1 - (A + I_{exc} + I_{inh})] + BI_{exc} \qquad (10)$$

To implement the inhibitory part, we need the I element with adaptive gain, or we can modulate the I element read beam by x_j to provide the multiplication. The second term is the excitatory component, which can be fed to the N element directly. For the steady-state case, the membrane potential is

$$x_j = \frac{BI_{exc}}{A + I_{exc} + I_{inh}} \qquad (11)$$

An optical implementation of pixel-by-pixel division was shown by Efron et. al. [11]; here we can use the output of the N element as the read beam of the I element to provide the required division.

References

[1] B. K. Jenkins and C. H. Wang, *J. of Opt. Soc. Amer. (A)* 4, 127A, paper PD6 (Dec. 1987); also "Model for an incoherent optical neuron that subtracts", *Optics Letters*, vol. 13, pp892-894, 1988.

[2] C. H. Wang and B. K. Jenkins, "Subtraction in optical neural network", *Proc. of the 1988 Connectionist Models Summer School*, pp513-522, Morgan Kaufmann Publishers, 1988.

[3] R. D. Te Kolste and C. C. Guest, "Optical competitive neural network with optical feedback", *Proc. IEEE First Int. Conf. on Neural Networks*, vol III, pp625-629, San Diego, 1987.

[4] K. M. Johnson, M. A. Handschy and L. A. Pagano-Stauffer, "Optical computing and image processing with ferroelectric liquid crystals", *Optical Engineering*, vol 26, No. 5, pp385-391, 1987.

[5] M. Wang and A. Freeman, "Neural function", *Little, Brown and Company* press, 1987.

[6] D. Psaltis, K. Wanger and D. Brady, "Learning in optical neural computing", *Proc. of IEEE first Int. Conf. on Neural Networks*, San Diego, vol III, pp549-556, 1987.

[7] D. Psaltis, X. G. Gu, and D. Brady, "Fractal sampling grids for holographic interconnections", *Proc. Optical Computering 88*, Toulon, France, J. W. Goodman, G. Robin, editors, Proc. SPIE 963, pp468-474, 1989. Nevada, OSA technical digest, vol 11, pp129-132, March 1987.

[8] G. A. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns", *Applied Optics*, vol 26, No. 23, pp4919-4930, Dec. 1987.

[9] S. Grossberg and E. Mingolla, "Neural dynamics of form perception: boundary completion, illusory figures, and neon color spreading", *Psychological Review*, vol 92, pp173-211, 1985.

[10] J. M. Wang, J. Taboury and P. Chavel, "Operating constraints of a light valve as a 2-D accumulating buffer for optical cellular processors", *Topical meeting on optical computing*, Salt Lake city, OSA technical digest, March 1989.

[11] U. Efron, E. Marom and B. H. Soffer, "Array division by optical computing", *Topical meeting on optical computing*, Incline Village, Nevada, OSA technical digest, 1985.

a reprint from Applied Optics

Subtracting incoherent optical neuron model: analysis, experiment, and applications

Chein-Hsun Wang and B. Keith Jenkins

To fully use the advantages of optics in optical neural networks, an incoherent optical neuron (ION) model is proposed. The main purpose of this model is to provide for the requisite subtraction of signals without the phase sensitivity of a fully coherent system and without the cumbrance of photon–electron conversion and electronic subtraction. The ION model can subtract inhibitory from excitatory neuron inputs by using two device responses. Functionally it accommodates positive and negative weights, excitatory and inhibitory inputs, non-negative neuron outputs, and can be used in a variety of neural network models. This technique can implement conventional inner-product neuron units and Grossberg's mass action law neuron units. Some implementation considerations, such as the effect of nonlinearities on device response, noise, and fan-in/fan-out capability, are discussed and simulated by computer. An experimental demonstration of optical excitation and inhibition on a 2-D array of neuron units using a single Hughes liquid crystal light valve is also reported.

I. Introduction

The potential advantages of using optics in the implementation of neural networks are well known and stem from the capability of optics for 3-D, high density interconnections and analog data storage, as well as rapid multiplication and addition of analog signals. As an example, consider a neural computation performed on a digital electronic machine. Figure 1 shows a sample procedure to compute the weighted sum of the membrane potential of a neuron based on a single digital processor. We see that over half of the time (55%) is spent on moving data and adjusting the pointer. Although pipeline and multiprocessor techniques can be used to speed up the computation, bottlenecks still exist in moving data between memories and registers. For N fully connected neurons, $O(N^2)$ multiplications and summations are required. If N processors are used so that each processor corresponds to one neuron, the computation time is $O(N)$ at best. A fully parallel analog system can do this in $O(1)$ time. In the electronic case the partitioning of the computation in hardware also causes limitations. This results in a trade-off that depends on the computation overhead, hardware complexity, power dissipation, and speedup. Because of these factors, it is pertinent to explore the use of analog optics, which has the potential of overcoming these scaleup problems. Analog optical processing accuracy is acceptable in many neural network applications. In addition, for pattern recognition and machine vision tasks, the inputs are light intensity. Optical neural networks can potentially process these inputs directly without any serial electronic conversion, increasing the likelihood of a fast, efficient system.

There are four main arithmetic operations in a conventional neural network: multiplication, addition, subtraction, and nonlinear thresholding. The optics can provide analog multiplication and addition in real time, while nonlinear thresholding can be performed by an optical modulator. Implementation of subtraction in an optical neural network is a key issue. Coherent and incoherent techniques for subtraction differ markedly.

A fully coherent optical system can subtract signals directly, using differences in the phase, or path length, of the optical beams. An example of such a system is in Ref. 1. Subtraction in this type of system is very efficient; the trade-off is that the system must keep relative path lengths stable to within much less than one wavelength. In addition, the phases of components in the system must be accurately controlled.

An incoherent system is more robust in terms of stability, position accuracy requirements, and noise

The authors are with University of Southern California, Department of Electrical Engineering, Signal & Image Processing Institute, Los Angeles, California 90089-0272.

Received 28 June 1989.
0003-6935/90/142171-16$02.00/0.

immunity. Signals are typically encoded as light intensities; this provides real, non-negative quantities which must at some point be subtracted. A variety of techniques for linear incoherent optical subtraction have been demonstrated by others.[2,3] Many of these techniques result in an absolute value of the difference $|Y - X|$, where Y and X are the image operands. The technique described in Ref. 3 uses a liquid crystal light valve (LCLV) and results in the difference image added to a constant bias image, i.e., $Y - X + B$, where B is a constant bias. Here we modify and extend this concept to provide directly a nonlinear function of a linear subtraction, which includes a threshold below zero and above some user-defined value (per neuron model). In this case, there is no need for an output bias, and in our model there is none; this enables direct cascadability as required in a neural net.

Figure 2 shows a paradigm for an optical neural network, which uses incoherent optical neurons combined with optical interconnections. Here, incoherent means the phase of the input light to the neuron is not being used for subtraction; the neuron outputs may still be coherent light. A hologram or holograms can be used to emulate synaptic weights in the optical neural network. The interconnection network should be adaptive during the training phase. Psaltis *et al.*,[4,5] for example, have discussed several learning and recalling issues in photorefractive crystals. As shown in Fig. 2, the incoherent optical neurons process the weighted sums from the interconnection network; they may also serve as an input transducer. The inputs of the neurons are also fed to the interconnection network to form correlations with the outputs of the neurons during the learning phase. Then the modified interconnection strength is stored in the interconnection network.

References 6 and 7 give examples of previous incoherent optical neural networks. Farhat and Psaltis *et al.*[6] used a hybrid electronic–optical scheme to implement a Hopfield net. Recently, Shariv and Friesem[7] demonstrated an all-optical neural network with only inhibitory neurons for the case of a Hopfield type network.

The objective of this paper is to present a model of an incoherent optical neuron (ION) for general optical neural networks and to assess its practicality. Its practicality is assessed via computer simulations of the ION in a neural network and via an experimental demonstration of an array of IONs using a liquid crystal light valve.

Section II reviews several existing techniques for incoherent subtraction utilizing an input and/or weight bias, in some cases coupled with complementary weights or inputs. These techniques suffer from bias buildup and/or thresholds that must vary from neuron to neuron. A technique described by TeKolste and Guest[8] eliminates these problems for the case of fully connected networks. Section III describes the ION model. Its uniform, fixed bias decreases the hardware complexity over previous techniques. It can be used to implement a variety of neuron models,

```
  calc()
register  a;        /* weight */
register  b;        /* input_value */
register  c;        /* store weighted sum */
register  d=n;      /* count index */
          c=0;
   while  (d=0)     (7 or 3)
          {
          a=        weight[d]; (4)
          b=        input_value[d]; (4)
          a=        a * b; (9)
          c=        c + a; (2)
          d=        d - 1; (2)
          }
          end;
```

Fig. 1. Sample procedure to calculate membrane potential based on a uniprocessor. The value shown in parenthesis is the number of clock cycles for an Intel 80386 processor. Clock period is 50 ns. Only 45% of the time is used in actual computation.

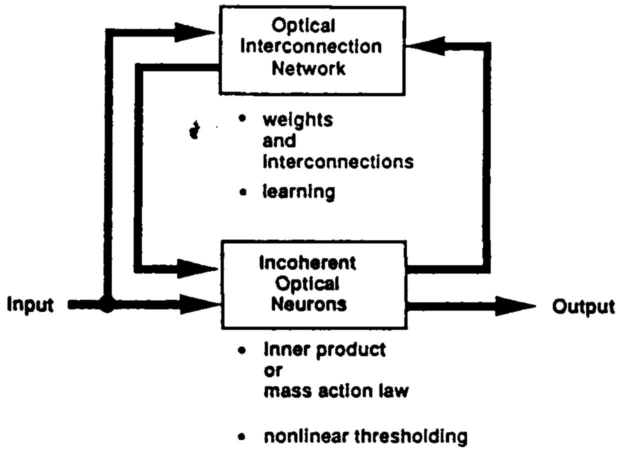

Optical Interconnection Network
• weights and interconnections
• learning
Input
Incoherent Optical Neurons
Output
• inner product or mass action law
• nonlinear thresholding

Fig. 2. Paradigm for an optical neural network.

including the binary neuron (e.g., McCulloch-Pitts, 1943[9]), and analog neuron (e.g., Grossberg, 1973,[10] Fukushima, 1975[11]).

Section IV discusses device and system imperfections that might affect the operation of the ION model, including undesired nonlinearities in the device response, as well as device and system noise. The immunity of the ION model to these imperfections is analyzed via computer simulations of it in a competitive neural network in Sec. V. Section VI reports on an experimental demonstration of incoherent subtraction using a Hughes liquid crystal light valve to implement a 2-D array of IONs. Section VII discusses a variant of the ION model, the linear subtraction ION. A special case of this model is used in the Amari[12]/Hopfield[13] net. Applications of the ION model in two kinds of neural network are discussed in Sec. VIII, i.e., Fukushima's multilayered networks[11,14-18] and a version of Grossberg's shunting networks.[19-23] A device requirement analysis for the ION is detailed in the Appendix.

II. Methods for Incoherent Subtraction

We model the recall or computation process of a single layer of a neural network as

$$\hat{V}_i = \psi\left(\sum_{j=1}^{N} W_{ij}V_j\right), \qquad (1)$$

where $\hat{V}_i$ is the output of neuron i, V_j are the signal inputs, W_{ij} are the synaptic weights, and $\psi(\cdot)$ is the output nonlinear function of the neuron, which is a nondecreasing function of the neuron inputs and has a finite range. Generally, W_{ij} can be positive, zero, or negative; $\hat{V}_i$ and V_j are non-negative in many neuron models but can take on negative values in other models.

In this section we review several methods of subtraction in incoherent optical neurons. Conceptually, the interpretation of an inhibitory signal can be either positive weight/negative signal or negative weight/positive signal. Of course, a bias can be added to either the input signals or the weights, yielding subtraction in a straightforward manner. First, the weight-bias method uses negative weights and positive outputs to code the inhibitory signals. The output of the neuron i is

$$\hat{V}_i = \psi\left[\sum_{j=1}^{N}(W_{ij} + W_b)V_j\right]$$

$$= \psi\left[\sum_{j=1}^{N} W_{ij}V_j + W_b\sum_{j=1}^{N} V_j\right], \qquad (2)$$

where W_{ij} represents the connection strength from the jth neuron to the ith neuron, which can be either positive or negative and is normalized to be between -0.5 and 0.5. When W_{ij} is positive, it represents an excitatory connection; when it is negative, it is inhibitory and is subtracted from the excitatory inputs. The term W_b is the bias in weight, usually 0.5, and V_j is the output of the jth neuron, which is positive and is between 0 and 1. The non-negative values $(W_{ij} + W_b)$ and V_j can then be implemented using incoherent optics. The second term $W_b\Sigma_{j=1}^{N} V_j$ is an input dependent bias, which is impractical for realization due to the required dynamic threshold of the neurons.

The second technique biases the input signals and uses positive weights and negative inputs for inhibitory signals and can be written as

$$\hat{V}_i = \psi\left[\sum_{j=1}^{N} W_{ij}(V_j + V_b)\right]$$

$$= \psi\left[\sum_{j=1}^{N} W_{ij}V_j + V_b\sum_{j=1}^{N} W_{ij}\right]. \qquad (3)$$

In this case the non-negative quantities W_{ij} and $(V_j + V_b)$ are represented physically using incoherent optics. The second term $V_b\Sigma_{j=1}^{N} W_{ij}$ is a weight dependent bias term. Since the weights of the neural network are changed from time to time to adapt to their environment by learning, this term is difficult to implement. We need to calculate the sum of weights into each neuron, which increases the implementation complexity, particularly if volume holograms are used. Nevertheless, this intensity bias method can potentially be used in the special case of neural networks that assume conservation of the sum of weights into a neuron. Such networks have been described, for example, by von der Malsburg.[24] In this case the second term $V_b\Sigma_{j=1}^{N} W_{ij}$ is a constant and can be treated as a fixed threshold of the optical neuron.

A similar technique called bias subtraction has been discussed by Gmitro and Gindi.[25] In their approach, the output of the neuron is positive and the weight can be either positive or negative to represent an excitatory or inhibitory connection efficiency. During implementation, two channels are used to process positive and negative weights separately; the negative channel is implemented by positive weights and complementary input signals, which are also positive:

$$\hat{V}_i = \psi\left[\sum_k W_{ik}V_k + \sum_j W_{ij}(1 - V_j)\right]$$

$$= \psi\left(\sum_k W_{ik}V_k - \sum_j W_{ij}V_j + \sum_j W_{ij}\right), \qquad (4)$$

where W_{ik} and W_{ij} are the weights to the positive (excitatory) and negative (inhibitory) inputs of the ith neuron. They are positive quantities during optical implementation and are represented by the transmittance of a mask or diffraction efficiency of a hologram. The bias term $\Sigma_j W_{ij}$ must be canceled out by adjusting the threshold of each neuron. This increases implementation complexity.

Another technique, proposed by TeKolste and Guest,[8] is a hybrid of the previous two:

$$\hat{V}_i = \psi\left[\sum_{j=1}^{N}\left(W_{ij} + \frac{1}{2}\right)V_j + \sum_{j=1}^{N}\frac{1}{2}(1 - V_j)\right]$$

$$= \psi\left[\sum_{j=1}^{N} W_{ij}V_j + \frac{N}{2}\right]. \qquad (5)$$

The bias term $N/2$ is independent of input as well as weight, which simplifies the requisite hardware considerably. This approach applies to unipolar neurons and fully connected networks only.

III. Incoherent Optical Neuron (ION) Model

From a biological point of view, the inhibitory site can be treated as a distinct mechanism from the excitatory site, e.g., neurotransmitters vs chemical-selected receptors.[27-29] The inhibitory site then accepts a positive input to produce a negative effect on the membrane of the neuron. The ION model follows this approach; it uses spatially and physically distinct control mechanisms to emulate the excitatory and inhibitory signal processing in a biological neuron.

The ION comprises two elements: an inhibitory (I) element and a nonlinear output (N) element. The inhibitory element provides an inversion of the sum of the inhibitory signals; the nonlinear element operates

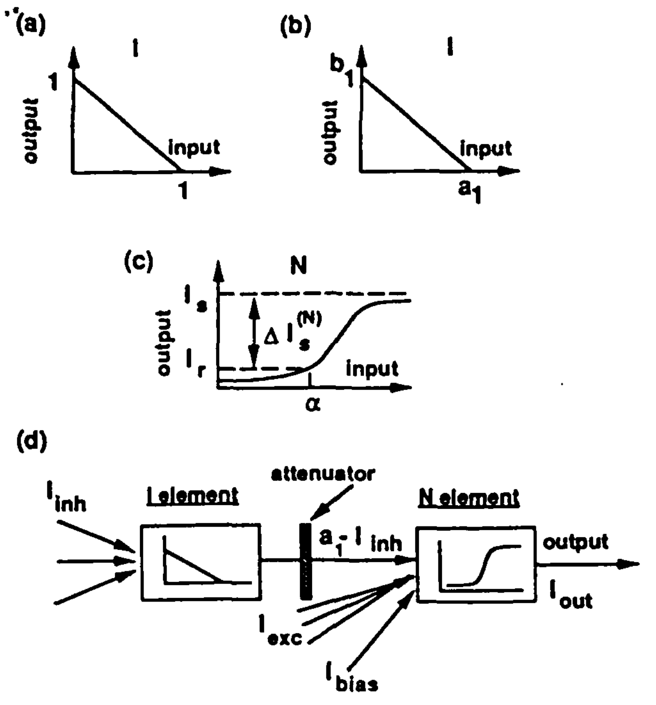


Fig. 3. The ION model: (a) normalized inhibitory (*I*) element; (b) unnormalized *I* element; (c) nonlinear (*N*) element; and (d) the ION structure.

on the excitatory signals, the inhibitory element output, and an optical bias to produce the output. The inhibitory element is linear; the nonlinear threshold of the neuron is provided entirely by the *nonlinear* output device. Figures 3(a)–(c) show the characteristic curves of the *I* and *N* elements, respectively. The structure of the ION model is illustrated in Fig. 3(d). The output of the normalized *I* element is given by

$$I_{\text{out}}^{(I)} = 1 - I_{\text{inh}} \tag{6}$$

and of the *N* element is given by

$$I_{\text{out}}^{(N)} = \psi[I_{\text{out}}^{(I)} + I_{\text{exc}} + I_{\text{bias}} - \alpha], \tag{7}$$

where I_{inh} and I_{exc} represent the total (weighted) inhibitory and excitatory neuron inputs, respectively. These include any lateral feedback signals as well as inputs from other layers. I_{bias} is the bias term for the *N* element, which can be varied to change the threshold, and α is the offset of the characteristic curve of the *N* element (refer to the Appendix for a more detailed discussion of α). The term $\psi(\cdot)$ denotes the nonlinear output function of the neuron. If we choose I_{bias} to be $\alpha - 1$, the output of the *N* element is

$$I_{\text{out}}^{(N)} = \psi(I_{\text{exc}} - I_{\text{inh}}), \tag{8}$$

which is the desired subtraction.

In general, the *I* element will not be normalized [Fig. 3(b)]. In this case the offset and slope of its response can be adjusted using I_{bias} and an attenuating element (ND filter), respectively, again enabling proper subtraction. (The unnormalized *I* element must have

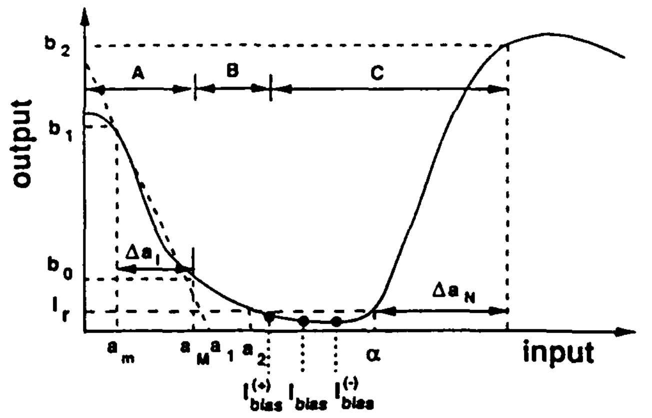


Fig. 4. Homogeneous case example: typical characteristic of a Hughes liquid crystal light valve serving as both *I* and *N* elements. Regions *A* and *C* are used for the *I* and *N* elements, respectively. Region *B* serves to separate the *I* element from the *N* element.

gain ≥1.) More quantitatively, the characteristic curve of the unnormalized *I* and *N* elements [Figs. 3(b) and (c)] can be modeled as

$$\begin{aligned} I_{\text{out}}^{(I)} &= -\frac{b_1}{a_1} I_{\text{inh}} + b_1 \quad \text{for } 0 \le I_{\text{inh}} \le a_1, \\ I_{\text{out}}^{(N)} &= \psi(I_{\text{in}}^{(N)} - \alpha) \quad \text{for } \alpha \le I_{\text{in}}, \end{aligned} \tag{9}$$

where $I_{\text{in}}^{(N)}$ denotes the sum of all inputs to the *N* element. The output of the *I* element is attenuated by an ND filter [Fig. 3(d)]; the intensity transmittance of the attenuator is equal to the magnitude of the slope of the characteristic curve of the *I* element. In this case, the bias for the *N* element is changed to $\alpha - a_1$.

The ION model can be implemented using separate devices for the *I* and *N* elements (heterogeneous case) or by using a single device with a nonmonotonic response to implement both elements (homogeneous case). Possible devices include bistable optical arrays[30–32] and spatial light modulators (SLMs) such as liquid crystal light valves (LCLV).[33,34] A single Hughes liquid crystal light valve can be used to implement both elements (Fig. 4).

A positive neuron threshold (θ) can be implemented in the ION by decreasing the bias by the same amount θ to $I_{\text{bias}}^{(+)}$ (Fig. 4). Similarly, a negative threshold is realized by increasing the bias by θ to $I_{\text{bias}}^{(-)}$. Distinct from the Gmitro and Gindi bias subtraction technique, this model inverts only the sum of inhibitory inputs to a neuron.

A detailed analysis for implementing the ION model is given in the Appendix, which describes the device requirements and threshold implementation as well as constraints on fan-in, fan-out, and gain. The maximum fan-in for the ION model is determined by the extinction ratio of the device, while the maximum fan-out is bounded by the device differential gain and system loss.

The features of the ION model include a bias that is essentially independent of input weight and signals, a dynamically and globally variable threshold, a capability of implementing a sigmoid or binary threshold function for different neuron models, cascadability and ease of implementation. Due to the separation of the control mechanisms in the ION model, it can implement more complex neuronal functions; for example, global inhibition by using the output of one inhibitory element to control the reading beam intensity of many excitatory elements.

IV. Effect of Device and System Imperfections

A. Nonlinearity in the *I* Element

Generally, the physical input/output characteristic of an I element will not be linear (see, e.g., Fig. 4). Subtraction can be obtained by further limiting the operation range of the I element to a sufficiently linear region. Let this region be $[a_m,a_M]$. In this case, the I element needs a bias a_m to operate in the linear region. Thus, the output of the I element can be modeled as

$$I_{\text{out}}^{(I)} = b_1 - \frac{b_1 - b_0}{\Delta a_I} I_{\text{inh}}, \tag{10}$$

where $\Delta a_I = a_M - a_m$ is the input range of the I element, I_{inh} is the inhibitory signal input, and $I_{\text{inh}} + a_m$ is the total intensity input to the I element. We define an effective slope of this pseudo linear region as $m^* = (b_1 - b_0)/\Delta a_I$, and then attenuate the output of the I element by m^*. The N element bias is set to be $\alpha - b_1/m^*$, and properly scaled subtraction is again obtained.

For example, for the biasing conditions of our LCLV (given in Sec. VI), if the inhibitory input is limited to the region $[0,0.6a_2]$, the normalized root mean square deviation from linearity (nmse) is reduced to 16% from approximately 36%. The quantity a_2 is defined as the input that is just sufficient to drive the output of the device to a functional 0 (see the Appendix). Here the normalized root mean square error is defined as

$$\text{nmse} = \frac{\sqrt{\int_{a_m}^{a_M} [y(x) - \hat{y}(x)]^2 dx}}{\sqrt{\int_{a_m}^{a_M} y^2(x) dx}}, \tag{11}$$

where $y(x)$ and $\hat{y}(x)$ are the ideal and actual outputs of the device with the input value of x, respectively. Simulations we have performed indicate that this amount of nonlinearity (16%) in the I element response is acceptable.

B. Noise Model for the ION

Here we use noise to mean any undesired signals, including perturbation of the operating point of the device, nonuniformity of the device, variation in operating characteristics from device to device due to production variation, internal noise inherent to the device, and environmental effects. Some of these effects are global (they affect all neuron units on a device identically), others are localized (each neuron unit behaves differently; the noise on neighboring neuron units on a device may be independent or correlated, depending on the source of the noise). Both temporal and spatial characteristics of the noise need to be included. The effect of noise on an additive lateral inhibitory network was discussed by Stirk *et al.*[35] Here, we construct a noise model for the ION by considering the origin and impact of the noise sources.

The possible noise sources in the ION model can be classified into four categories: input noise, device noise, system noise, and coupling noise. The input noise can be modeled at the device input and includes environmental background noise and residual output of the optical devices. Essentially, it has nonzero mean and varies slowly with time. The device noise is mainly caused by uncertainty in the device's characteristics, for example, drift of the operating point and variation of gain due to temperature or other effects. The system noise has a global effect on all neuron units on an optical device and includes fluctuations in the optical source. Finally, the coupling noise (crosstalk) is due to poor isolation between the optical neuron units, crosstalk from the interconnection network, and imperfect learning. As noted in Ref. 35, alignment inaccuracies and imperfect focusing and collimating optics also cause localized crosstalk. Coupling noise is signal dependent.

1. *Input Noise*

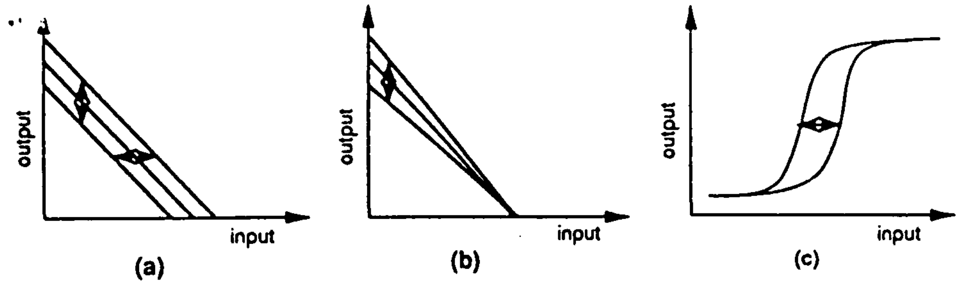
Let the environmental background noise for the I and N elements be denoted by $N_b^{(I)}$ and $N_b^{(N)}$, respectively. The total residual output noise N_r is caused by residual output of the optical device. At the input of an incoherent optical neuron it is $N_r = \Sigma_{j=1}^{N_{\text{in}}} W_{ij} I_r / N_{\text{out}}$, which is weight dependent and varies slowly with time due to learning. Term W_{ij} is the interconnection strength from neuron j to neuron i, I_r is the residual output of the optical device [Fig. 3(c)], and N_{out} denotes the fan-out of the optical neuron unit. Perturbation of the weights can be treated as an input dependent noise source as $N_w = \Sigma_{j=1}^{N_{\text{in}}} \Delta W_{ij} x_j$, where each ΔW_{ij} is assumed independent. For the interconnection network, imperfect learning of the weights, nonuniformity of the weights, residual weights after reprogramming, and perturbation of the reference beam intensity will cause weight noise. Since each of these noise sources can occur at the input of both I and N elements, the output of the ION for the case of normalized characteristics is

$$I_{\text{out}} = \psi(\{1 - [I_{\text{inh}} + N_b^{(I)} + N_r^{(I)} + N_w^{(I)}]\} + \{I_{\text{exc}} + N_b^{(N)} + N_r^{(N)} + N_w^{(N)}\} + [\alpha - 1] - \alpha). \tag{12}$$

If the background noise is space invariant and the I and N elements have the same device area, the terms $N_b^{(I)}$ and $N_b^{(N)}$ will cancel out. The residual noise terms $N_r^{(I)}$ and $N_r^{(N)}$ and weight noise terms $N_w^{(I)}$ and $N_w^{(N)}$ generally do not cancel.

2. *Device Noise*

We model two noise effects in the I element, as illustrated in Figs. 5(a) and (b): shift (drift) and gain variation in the device characteristics, which are denoted as $N_d^{(I)}$ and $N_g^{(I)}$, respectively. For the output N element, the gain variation [Fig. 5(e)] only modifies the



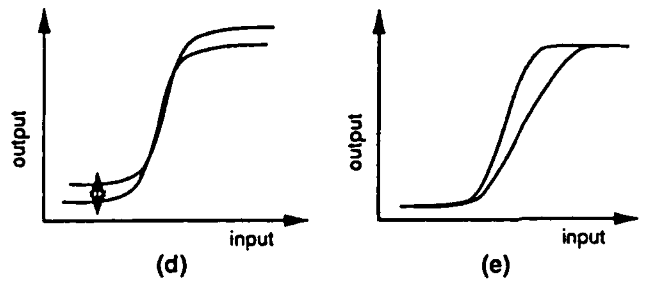


Fig. 5. Modeling the device noise of an incoherent optical neuron: I element with (a) drift (vertical/horizontal), (b) gain variation; N element with (c) horizontal drift or variation of the bias; (d) vertical drift, and (e) gain variation.

nonlinearity of the element N. If this gain variation is a slowly varying effect, it will have little effect on the dynamic behavior of the network; so for the N element we consider only drift which is denoted by $N_d^{(N)}$. The N element drift can be horizontal [$N_{\rm dh}^{(N)}$] [Fig. 5(c)] or vertical [Fig. 5(d)]. The vertical drift of one neuron unit becomes an additive noise at the input of the next neuron unit, and so will be approximated by including it in the residual noise term above. The horizontal drift has the same effect as a perturbation in the bias term, denoted by $N_{\rm bp}$.

If the gain variation is small, it can be included in the I element response by expressing its output as $(1 + N_g)(1 - I_{\rm inh})$, where N_g denotes the gain noise.

3. *System Noise*

System noise has a global effect on the ION. If it is caused by an uncertainty in the optical source, it causes a variation in the characteristic curve of the device and a perturbation in the bias term. In the case of an LCLV, a perturbation in the reading beam intensity produces a gain variation in the I element and a combination of gain variation and horizontal drift in the N element (or equivalently, essentially a rescaling of its output axis). Device gain variation of the I element was discussed above and is local, i.e., it varies from one neuron unit to another on a given device. Variation in gain due to system noise is global. The normalized device noise and system noise can be modeled as

$$I_{\rm out} = \psi\{[1 + N_g^{(I)}][1 + N_d^{(I)} - I_{\rm inh}] + I_{\rm exc} + N_{\rm dh}^{(N)} + (\alpha - 1 + N_{\rm bp}) - \alpha\}. \qquad (13)$$

4. *Crosstalk*

Crosstalk can be caused by the physical construction of the interconnection network [e.g., coupling between different holograms, diffraction in the detection (neuron unit inputs) plane, inaccurate alignment and focusing]. It can also be caused by imperfect learning or reprogramming of the synaptic weights, where the perturbation of different weights is correlated. In general, crosstalk is signal dependent and varies from one neuron unit to another on a given device. It can be modeled as an input noise to the I and N elements. It is excluded from our current simulations because it is signal dependent.

Based on the above discussion, these noise terms can be grouped into additive (N_I^{+}) and multiplicative (N_I^{*}) noise of the I element and additive noise (N_N^{+}) of the output element N. The general noise model of the ION can then be written as

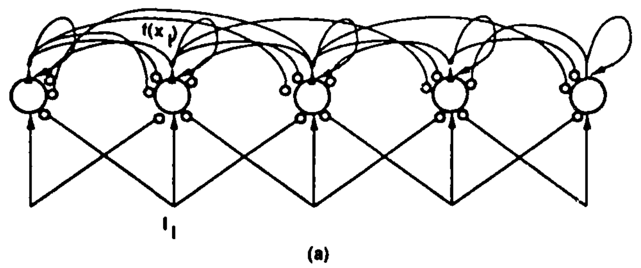
$$I_{\rm out} = \psi\{(1 + N_I^{*})[1 - I_{\rm inh} + N_I^{+}] + I_{\rm exc} + N_N^{+} - 1\}, \qquad (14)$$

where N_I^{+} is the sum of the drift noise [$N_d^{(I)}$], background noise [$N_b^{(I)}$], residual noise [$N_r^{(I)}$], weight noise [$N_w^{(I)}$], and crosstalk noise [$N_c^{(I)}$]; N_I^{*} is the gain noise of the I element; and N_N^{+} is the sum of the background [$N_b^{(N)}$] and residual input noise, horizontal shift noise [$N_{\rm dh}^{(N)}$], weight noise and crosstalk noise of the N element, and bias noise [$N_{\rm bp}$].

V. Computer Simulation

A. *I* Element Compensation

To assess the effect of imperfect device responses for the I element, we have performed simulations on a variant of Grossberg's on-center off-surround competitive network[10,26] for edge detection (Fig. 6). The network contains thirty inner-product type neurons connected in a ring structure with input and lateral on-center off-surround connections. To optimize the use of the (nonlinear) I element, an attenuator (neutral density filter) can be placed in front of the I element to



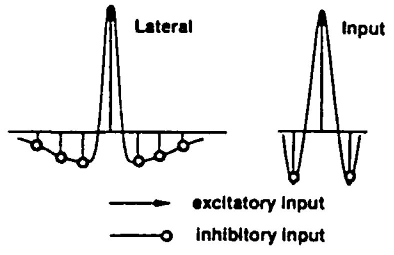


Fig. 6. On-center off-surround competitive neural network: (a) network and (b) interconnection weight strengths as a function of distance from a neuron. Interconnections in this network are space invariant.

reduce the overall gain to bring it closer to the ideal response. Figure 7 shows computer simulations of the network responses based on a nonlinear curve[36] that is a close approximation to our measured LCLV characteristic (Sec. VI) for different input attenuations. Each resolvable row of Fig. 7 represents a 1-D simulation on a distinct 1-D input. Thirty different binary inputs were each simulated at four different input signal levels. Apparently the attenuation can have a tolerance of approximately ±20%. In our simulations we used an attenuator but no input bias to the I element; the region of operation for these curves extended over most of the input range of the device ($[0, 0.7a_2]$).

B. Noise Effect

We use the same network to test the effect of noise (of course the results are actually network dependent) to get an idea of the noise immunity and robustness to physical imperfections of the ION model. In the computer simulation, each of the three noise sources in Eq. (14) are assumed independently distributed Gaussian with zero mean. We define the maximum perturbation p of the noise source as twice the standard deviation, expressed as a percentage of the input signal level. A normalized mean square error (nmse) [see Eq. (11)] is used to measure the acceptability of the result. Although it is not a perfect measure, a nmse < 0.1–0.15 generally looks acceptable for the network response due to our input test pattern.

The quality of the output is a function of the temporal and spatial correlation of the noise. Figure 8(a) shows the nmse vs percentage of maximum noise perturbation for the input level of 0.7 and for noise that is correlated over different time periods T. The noise sources for each neuron unit are assumed independent and identically distributed. The temporal correlation of each noise source with its previous values is given by $N(t+1) = \Sigma_{i=1}^{T} h_i N(t+1-i)$, and the correlation coefficients h_i decrease linearly with i (to $h_T = 0$). In Fig. 8(a), all three noise sources in our model are present and have the same variance. If the acceptance nmse criterion is 0.15, a perturbation of ±10% on each noise source yields an acceptable result in all cases. The nmse increases as the input level and noise variance increase [shown in Fig. 8(b) for $T = 50$].

In some cases, the noise is spatially correlated. We simulated the network with spatially correlated noise. The spatial correlation is assumed to have a Gaussian profile. Figures 9(a) and (b) are the responses for a spatial correlation range of 3 and 13, respectively, while Figs. 9(c) and (d) show the responses for spatially and temporally correlated noise.

Drift of the device characteristic is a global effect. Figure 10 simulates slowly varying and quickly varying I and N element drifts on this network; a ±10–15% perturbation in drift is apparently acceptable. Figure 11 shows the effect of local I element gain variation that is spatially correlated; a ± 10–15% perturbation in gain is apparently acceptable.

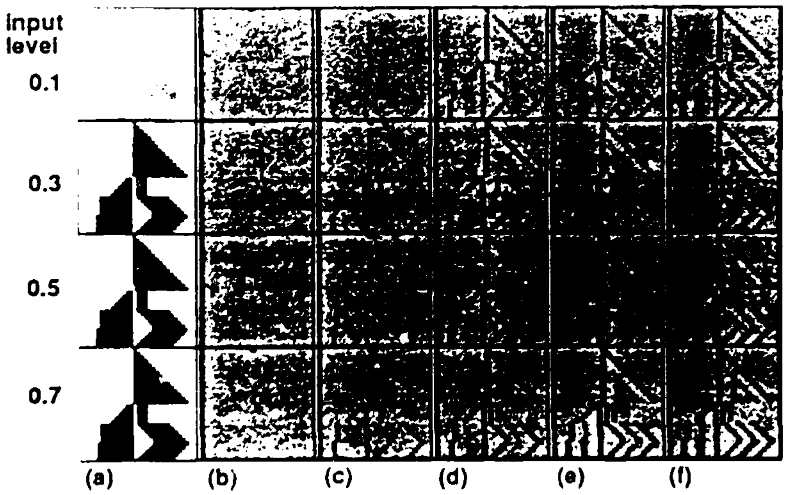
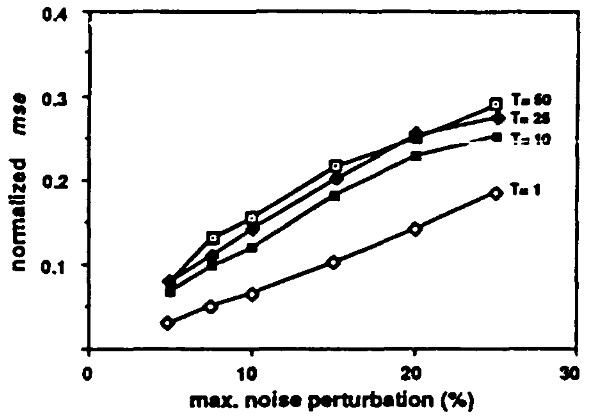


Fig. 7. Network responses for different attenuation factors, s_1, at the input to the nonlinear inhibitory element. Each horizontal scan line of the figure represents a separate simulation on a 1-D input: (a) input pattern; (b) $s_1 = 1.0$, no attenuation; (c) $s_1 = 1.5$; (d) $s_1 = 2.5$; (e) $s_1 = 3.0$; and (f) $s_1 = 3.5$. The ideal output is essentially identical to (d).

(a)



(b)

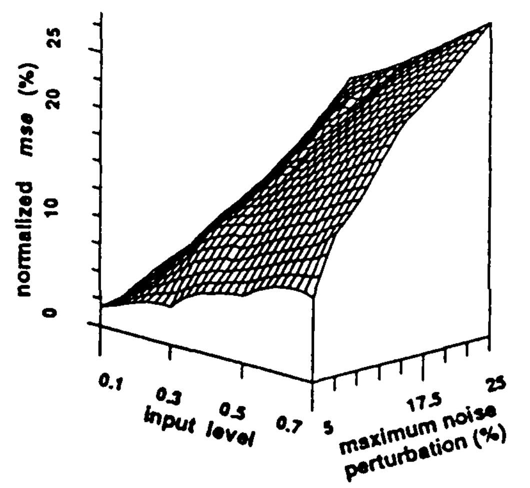


Fig. 8. Normalized mean square error (nmse) measure of the network response for temporally correlated noise. Three noise sources, N_I^+, N_I^-, and N_N^+ are simulated. (a) Normalized mean square error of the net output vs maximum noise perturbation p for correlation periods (T) ranging from 1 to 50. The input level is 0.7. (b) Output nmse plot for different noise perturbations and input levels ($T = 50$).

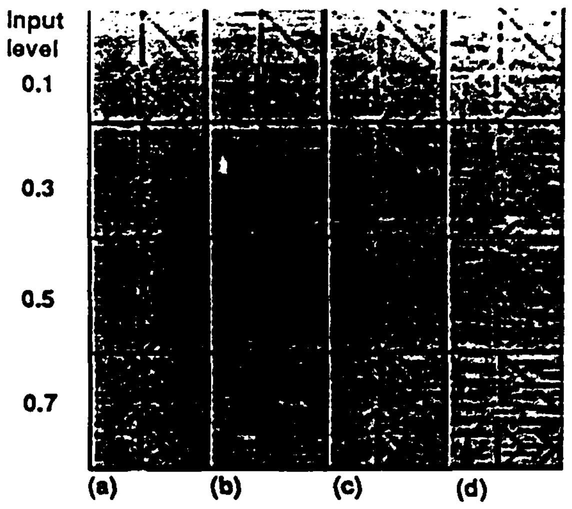


Fig. 9. Simulation results for spatially and temporally correlated noise; sc is the spatial correlation range and T is the temporal correlation period. Three noise sources are simulated simultaneously, each with perturbation $p = \pm 10\%$. The nmse is the normalized mean square error of the network output. Only spatially correlated noise ($T = 1$) with (a) $sc = 3$, nmse = 0.08, (b) $sc = 13$, nmse = 0.13; spatially and temporally correlated noise ($T = 25$) with (c) $sc = 3$, nmse = 0.14, (d) $sc = 13$, nmse = 0.18.

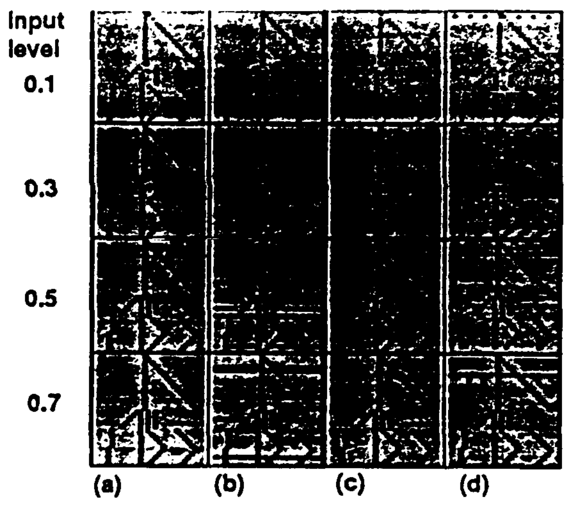


Fig. 10. Effect of device drift in the ION. The drift is uniform over all neuron units. High frequency drift ($T = 1$) with (a) $p = \pm 10\%$, nmse = 0.02, (b) $p = \pm 25\%$, nmse = 0.09; low frequency drift ($T = 50$) with (c) $p = \pm 10\%$, nmse = 0.07, (d) $p = \pm 25\%$, nmse = 0.11.

VI. Experiment

Figure 12 shows the experimental setup for implementing and testing an array of IONs. Three input beams are used to provide N element bias, I element inputs, and N element inputs, which are controlled by polarizer pairs $P1$, $P2$, and $P3$, respectively. The I element input path, SI-$BS5$-$BS7$-$L2$-$BS8$-LCLV, is imaged with a magnification factor of 0.8. The same magnification factor is applied to the N element input path, SN-$BS7$-$L2$-$BS8$-LCLV. Two feedback paths are implemented, one for the I to N connection, which is $BS9$-$BS10$-$L3$-$BS11$-$L5$-$BS12$-$BS8$-LCLV. The other feedback path is through mirrors $M5$ and $M6$, and is for the N to N self-feedback connection. Each feedback path images from the LCLV output plane to the LCLV input plane; the I to N feedback path also shifts the image to the N element input. Mask $MK2$ is used to block the bias beam to the I elements. Masks $MK3$ and $MK4$ block the N and I element outputs in the I to N and N to N feedback paths, respectively.

Figures 13(a) and (b) give the input/output characteristics of the I and N elements for an applied voltage

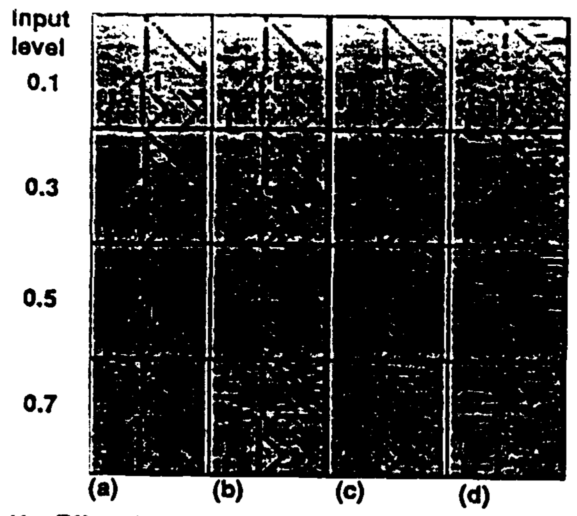


Fig. 11. Effect of gain variation of the I element in the ION. This effect is nonuniform with some spatial correlation. High frequency gain variation with (a) $T = 1, sc = 3, p = \pm 10\%$, nmse $= 0.04$, (b) $T = 1, sc = 3, p = \pm 25\%$, nmse $= 0.11$; low frequency variation with (c) $T = 25, sc = 9, p = \pm 10\%$, nmse $= 0.09$, (d) $T = 25, sc = 9, p = \pm 25\%$, nmse $= 0.18$.

of 5.0 V rms at a frequency of 1.5 kHz. In our case, the Hughes LCLV used has a twisted nematic liquid crystal and a CdS photoconductor. Self-feedback for the N element is necessary to fulfill the constraints of the ION model. Figure 14 shows experimental results of a single neuron. Figures 14(a)–(d) show binary subtraction, while Figs. 14(e) and (f) show gray level subtraction. The neuron size is ~2-mm diameter, as was the pixel size for the measurements in Fig. 13.

To demonstrate a 2-D array of neuron units, the continuous case was implemented, i.e., no isolation between neuron units. Figures 15(a) and (b) show the experimental result for binary subtraction. Two character sets, each 6-mm square, are chosen for the N (left side) and I (right side) inputs [Fig. 15(a)]. All four possible cases are included (corresponding to 1 or 0 for the N element input and 1 or 0 for the I element input). A bias is added to the N element inputs as described in Sec. III. Figure 15(b) shows the I element outputs (right side) and the final neuron outputs (left side; these are also the N element outputs). The ideal result is the residual leg of the character R (right top) and the full character T (right bottom) in the N element area, and is in agreement with Fig. 15(b); these regions correspond to a 1 on the excitatory inputs and a 0 on the inhibitory inputs.

To test the subtraction linearity for the gray level case, we set the I inputs to their minimum and the N inputs to a small but nonzero value and measure the N element output. For every increment in the I input, we adjust the N input to keep the N element output constant. Figure 15(c) is a plot of the (normalized) resulting linear subtraction of the N inputs from I inputs. Essentially, it is equal to the complementary plot of the I element response. Here we use the full operation range of the inverting region of the LCLV. If we limit the operation of the I inputs to be 50% of a_2, we obtain subtraction that is very close to linear. In our setup, the range of the attenuated I output measured at the N element input is ~0.2 μW/cm^2. With the current setup, the nonlinearity of the N element needs to have relatively high differential gain to compensate for the significant loss in the feedback ($I \rightarrow N$) path, thus the use of self-feedback for the N elements.

VII. Variant of the ION Model

The ION model emulates a biological neuron by using spatial coding for the sign of the input signals. To accommodate some of the artificial neuron models that use bipolar neuron outputs, a variant of the ION model is proposed which uses complementary inputs and weights.[36] It is given by

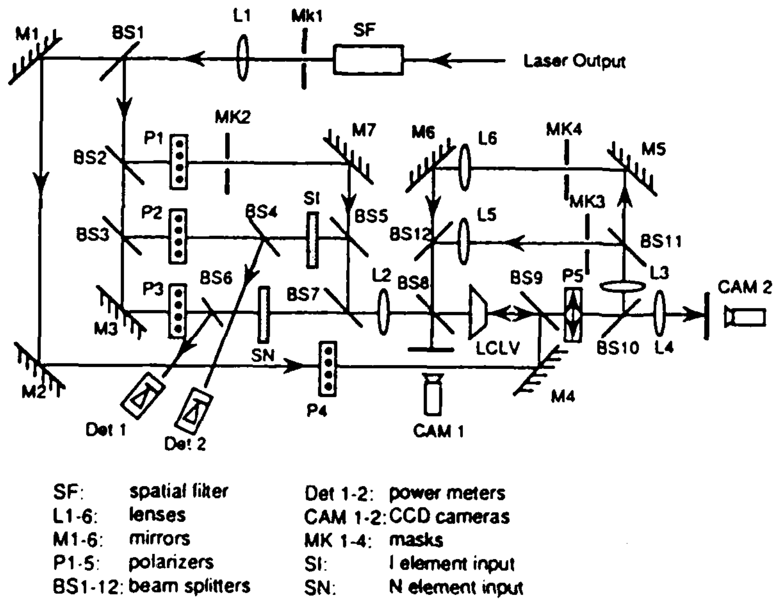


Fig. 12. Experimental setup of the ION test circuit.

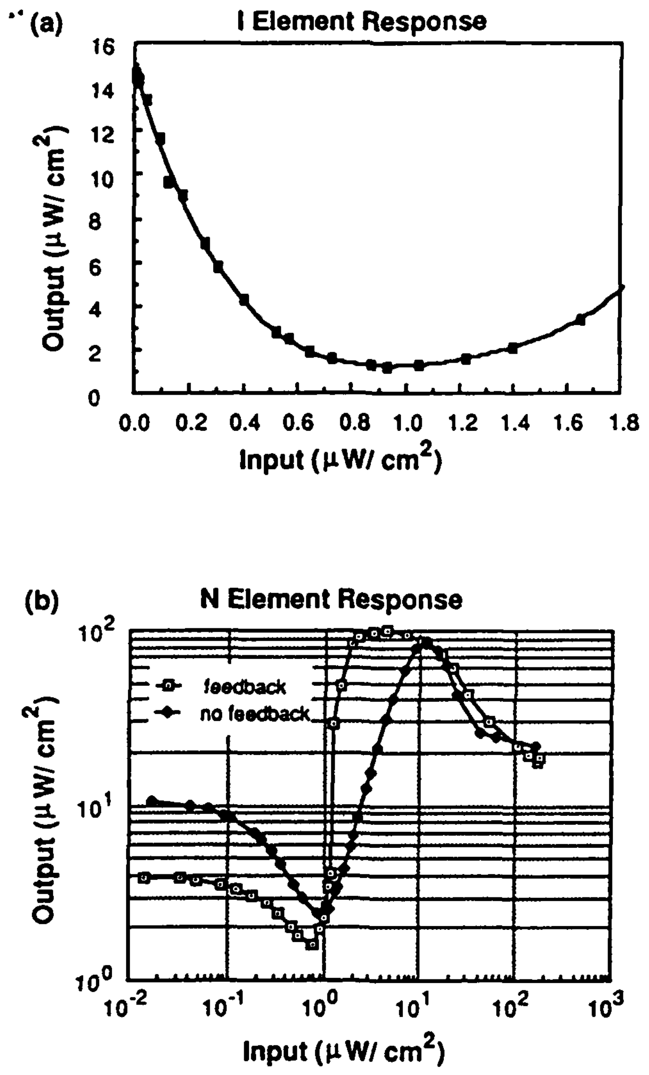


Fig. 13. LCLV characteristics of the I and N element in the test circuit for V = 5.0 volts, f = 1.5 kHz. The vertical axis is the intensity measured at the LCLV output, when in the system of Fig. 12 with a laser power of 200 mW. The I element is fairly linear within 50% of its operation range. The self-feedback of the N element (b) is necessary to satisfy the ION requirement for this particular device.

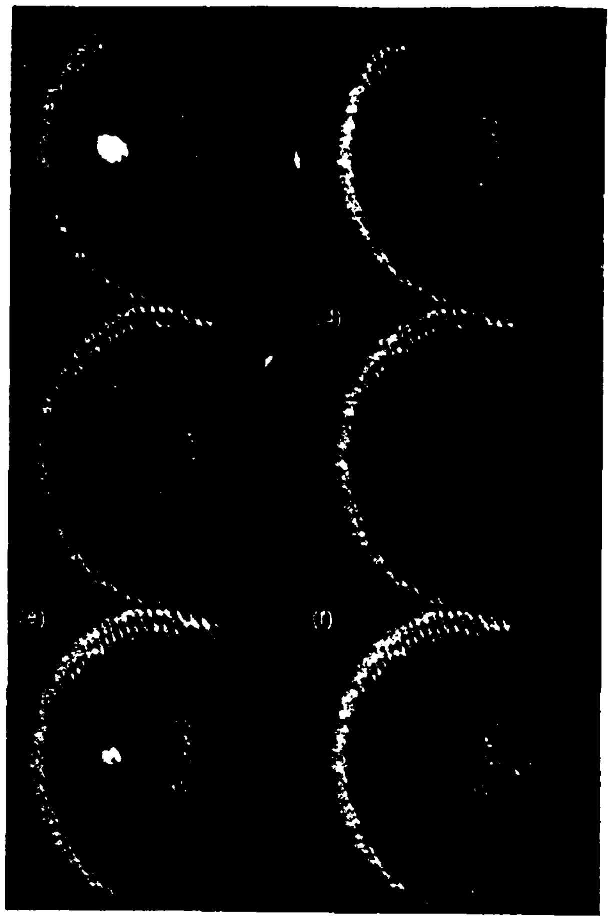


Fig. 14. Experimental results of a single neuron showing N element output (left side of each figure) and I element output (center pixel of each figure). The rightmost pixel is used for alignment. The following data are normalized: (a) $I_{exc} = 1, I_{inh} = 0$; (b) $I_{exc} = 1, I_{inh} = 1$; (c) $I_{exc} = 0, I_{inh} = 0$; (d) $I_{exc} = 0, I_{inh} = 1$; gray level case, with (e) $I_{exc} = 1, I_{inh} = 0.5$; and (f) $I_{exc} = 0.5, I_{inh} = 0.5$.

$$\frac{1}{2}(1+\hat{V}_i) = \psi\left[\sum_{j=1}^{N}\frac{(1-W_{ij})}{2}\,\frac{(1-V_j)}{2} + \sum_{j=1}^{N}\frac{(1+W_{ij})}{2}\,\frac{(1+V_j)}{2}\right]$$
$$= \psi\left[\sum_{j=1}^{N}\frac{W_{ij}V_j}{2} + \frac{N}{2}\right]. \qquad (15)$$

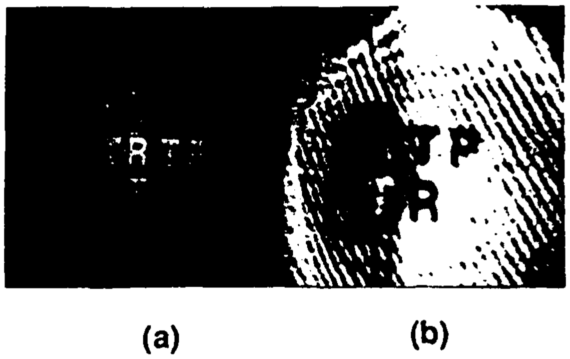
The terms $(1 - V_j)/2$, $(1 + V_j)/2$, $(1 - W_{ij})/2$, and $(1 + W_{ij})/2$ are positive. The $(1 + V_i)/2$ and $(1 - V_i)/2$ terms can be generated in some complementary devices by using orthogonal polarizations (e.g., LCLV) or by reflected and transmitted beams (e.g., some bistable optical devices); this makes the model directly cascadable. In this case, the input to the net must be inverted, and the signal and its complement are preserved throughout the network. No other I elements are necessary. If such complementary devices are not available, the neuron input and output can be left in the form $(1 + V_i)/2$. Then an I element can be used at each neuron to generate $(1 - V_i)/2$ from $(1 + V_i)/2$.

In this model, there is no spatial distinction between excitatory and inhibitory channels. Restricting the model to fully connected networks [as in Eq. (15)] keeps the threshold neuron independent; this yields the simplest hardware implementation. Alternatively, by summing in Eq. (15) only over the inputs to each neuron, a partially connected network can be implemented, at the expense of an increase in hardware complexity because of the neuron dependent threshold.

For example, an Amari net[12] uses binary bipolar neurons and its retrieval operation is given by

$$\hat{V}_i = \Phi\left[\sum_{j=1}^{N} W_{ij}V_j\right], \qquad (16)$$

where $\hat{V}_i \in \{+1,-1\}$ is the output state and $W_{ij} \in [-1,1]$ is the normalized weight from neuron j to neuron i.



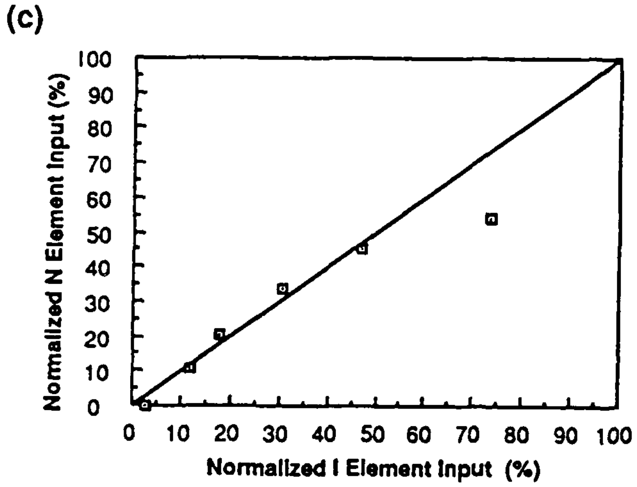


Fig. 15. Results of binary subtraction with (a) input patterns, N inputs (left) and I inputs (right), at LCLV input; and (b) outputs, N element outputs, the subtraction result (left), and I element outputs (right). The ideal result is the residual leg of R (top right) and full T (bottom right) of the N element output. (c) Gray-level subtraction showing normalized N input vs I input for a constant N output. The subtraction is quite linear if we only use 50% of the maximum I input (a_2).

The nonlinear output function $\Phi(x)$ is equal to 1 for $x \geq 0$, otherwise it is -1. The net is fully connected. Our complementary model can implement this directly.

VIII. Application Examples

As an example of the use of the original ION model in a neural net, a conceptual diagram of an implementation of a single layer feedback net is shown in Fig. 16. It utilizes a single 2-D spatial light modulator for both I and N elements. The output of the I element is imaged onto the input of the N element, after passing through a ND filter as the (uniform) attenuation. A uniform bias beam is also input to the N element. The N element output is fed back through an interconnection hologram to the inputs of both I and N elements, representing inhibitory and excitatory lateral connections, respectively.

Cooperative and competitive interactions[37,38] are two main neural mechanisms of information processing in the human brain. The macroscopic behavior of the neural network exhibits a cooperative property but it may locally execute competitive operations. Structurally, these two mechanisms are comprised of interactions of excitatory and inhibitory signals through feedforward, lateral, and feedback connections in the neural network. Heretofore, the discussion has considered only conventional inner-product neurons, which perform a weighted sum of their input signals. The ION concept can also be applied to other types of neuron, for example, that based on mass action. These neurons use a mass action law to model neuron behavior,[26] which tends to cause competition for the limited membrane sites. The following will briefly discuss two types of neural network models, those of Fukushima and Grossberg, and their implementation using ION.

Competitive neural networks can be used in feature extraction, pattern recognition, and associative memory.[14-17;19-23] Fukushima's neocognitron is a multilayer feedforward neural network used for pattern recognition. His more recent models are bidirectional and can serve as feature extractor, pattern recognizer, and associative memory.[15-18] The interaction relationship of his models can be written as

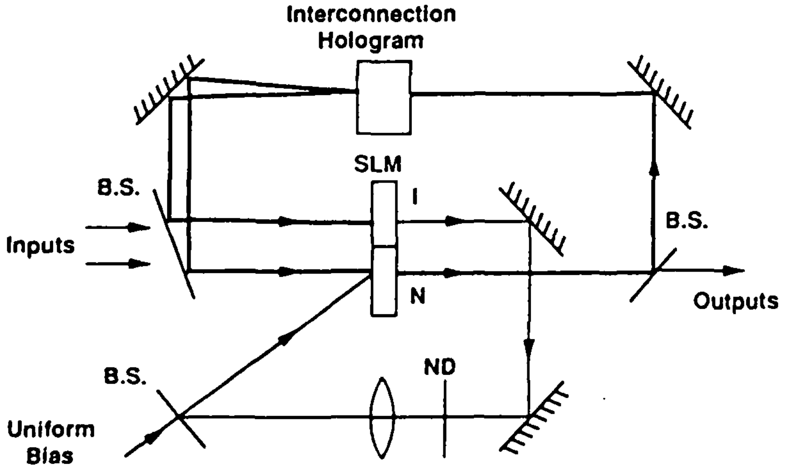


Fig. 16. Single layer feedback net using a single spatial light modulator to implement both I and N elements.

$$V_i = \psi\left\{\sum_{j\in A_{\rm pre}} a_{ij}V_j + \sum_{k\in A_{\rm post}} a_{ik}V_k - d_i \sum_{j\in A_{\rm pre}} \tilde{c}_{ij}V_j - b_i \sum_{k\in A_{\rm post}} \tilde{c}_{ik}V_k - \sum_{l\in A_l} \tilde{e}_{il}V_l\right\}, \tag{17}$$

where tilded weights ($\tilde{c}_{ij}$, $\tilde{c}_{ik}$, and $\tilde{e}_{il}$) are fixed weights with a Gaussian distribution with respect to the distance between the current cell and the input cell. Terms $A_{\rm pre}$, $A_{\rm post}$, and A_l denote the interconnection region from the previous layer, postlayer, and lateral inhibition area, respectively. The other weights (a_{ij}, a_{ik}, b_i, and d_i) are modifiable subject to winner take all learning, i.e., within a specified region, only the neuron with the strongest output can modify its weights. The first two terms in Eq. (17) are excitatory inputs from previous and postlayers, and the third and fourth terms are used to provide adaptive level control. The last term in Eq. (17) is the lateral inhibition (i.e., inhibitory connections within the layer). The ION model can implement these by putting the last three terms into the I element, with the first two terms going directly to the N element. Typically, only a subset of these terms is present in any one of Fukushima's models; for example, the cognitron and neocognitron models[11,14] use only the first, third, and fifth terms, while his hierarchical associative memory[16] uses all but the last term.

Another cooperative competitive type of neural network is Grossberg's on-center off-surround enhancement and adaptive resonance theory.[19–23] Grossberg uses a mass action law in his models, which can be described as

$$\begin{aligned} \dot{x}_i &= -Ax_i + (B - x_i)I_{\rm exc} - x_i I_{\rm inh}, \\ I_{\rm exc} &= \sum_{j=1}^{n_1} \psi(x_j)C_{ij} + I_i, \\ I_{\rm inh} &= \sum_{j=1}^{n_2} \phi(x_j)D_{ij} + J_i, \end{aligned} \tag{18}$$

where x_i is the membrane potential of neuron i, and $I_{\rm exc}$ and $I_{\rm inh}$ denote the total excitatory and inhibitory inputs to neuron i. Terms $\psi(x_j)$ and $\phi(x_j)$ are the output of neuron j and its inhibitory interneuron, respectively, and are sigmoid functions in most cases. Terms I_i and J_i are the total excitatory and inhibitory inputs from other layers. Terms A and B are the decay constant and maximum membrane potential, respectively; C_{ij} and D_{ij} are the interconnection weights within this layer. The above equation can be grouped into two terms to be implemented by I and N elements, respectively. For the discrete case, we can rewrite Eq. (18) as

$$x_i(k+1) = x_i(k)[1 - (A + I_{\rm exc} + I_{\rm inh})] + BI_{\rm exc}. \tag{19}$$

This is a shunting model. The crucial difference from an implementation point of view is the product between the neuron inputs and the neuron potential in the first term of Eq. (19). To implement this term, an I element with adaptive gain is needed; or the I element read beam can be modulated by x_i to provide the multiplication. The second term is the excitatory part, which can be fed to the N element directly. For the lumped case of Grossberg's model, there is no delay between the output of the neuron and its interneurons. We assume that the output characteristic functions $\psi(\cdot)$ and $\phi(\cdot)$ are the same to simplify the complexity of the neuron. For the steady state case, the membrane potential is

$$x_i = \frac{BI_{\rm exc}}{A + I_{\rm exc} + I_{\rm inh}}. \tag{20}$$

An optical implementation of pixel-by-pixel division has been shown by Efron *et al.*[39]; here we can use the output of the N element as the read beam of the I element to provide the required division. Figure 17 shows a conceptual diagram to implement Grossberg's cooperative competitive network. In the figure, we use $\psi(x)$ to approximate the membrane potential x. (This only holds for output characteristics that are nearly linear. If this is not the case, the N elements shown are linear, and an additional nonlinear N element device is inserted before the interconnection hologram.) In Fig. 17, masks B and C are used to provide read beams for the N and I elements, respectively. Each mask is in an image plane of the LCLV. The read beam of the I element derives from the N element output through mask C. Here we only show the external input and lateral on-center off-surround connections, which are realized by interconnection unit F. The most powerful interconnection unit would utilize holograms in a volume medium.

For the unlumped system, which is more common in the biological neural network, the inhibitory signal comes from an interneuron with a different characteristic. The interaction can be formulated as

$$\begin{aligned} \dot{x}_i &= -Ax_i + (B_i - x_i)\left[\sum_j^{n_1} \psi(x_j)C_{ij} + I_i\right] - x_i\left[\sum_j^{n_2} \phi(y_j)D_{ij} + J_i\right]; \\ \dot{y}_i &= -Ey_i + \sum_l^{n_3} x_l F_{il}, \end{aligned} \tag{21}$$

where y_i is the activation state (potential) of interneuron i, which receives excitatory signals from a total of n_3 excitatory neurons. Term F_{ij} represents the interconnection strength from neuron j to neuron i and D_{ij} is the weight from the output of the interneuron to its neighboring excitatory neurons. For the full neuron with potential $x_i(k)$, we need one I element with adaptive gain and one linear N element. The interneuron, which has potential $y_i(k)$, is implemented by one N element only.

IX. Discussion and Conclusion

A general model for optical neuron units has been discussed to perform the requisite subtraction optically in incoherent optical neural networks. This model uses two separate responses to implement an optical

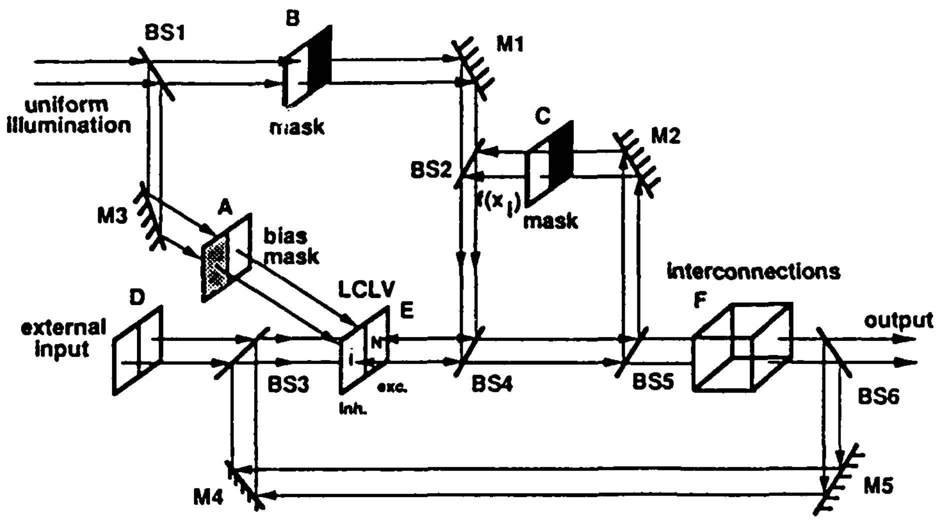


Fig. 17. Conceptual diagram to implement Grossberg's mass action type neuron based on the ION model utilizing a LCLV.

neuron with dynamic and global thresholding. The ION model can implement analog or binary non-negative neuron unit outputs, excitatory and inhibitory neuron unit inputs, and can be used in fully or partially connected neural networks. Implementation of conventional inner product neuron units and mass action law neuron units for shunting networks using the ION concept has been described.

In addition, a variant of the ION was given that encodes signals and their analog complements in a two-channel system. It permits bipolar neuron outputs. The trade-off is an increase in interconnection network complexity and the requirement for a neuron dependent threshold for the case of partially connected networks.

We have summarized sources of noise for the ION and proposed a noise model. From the result of computer simulations, it seems that the example network performs much better for quickly varying (i.e., temporally uncorrelated) noise than for temporally correlated (more slowly varying) noise. Due to the static input pattern and the competitive nature of the network, once the noise term has survived a number of iterations, it will continue to get stronger and will not die out. For other neural networks, if the input to the network is time varying, slowly varying noise is effectively an offset response of the network and might be adaptively overcome by the network, while quickly varying noise interacts with the input patterns and is generally more difficult to compensate.

For noise that is correlated, we have found that the qualitative effect of each of the three noise sources (additive inhibitory, multiplicative inhibitory, and additive excitatory) on the output of the net is essentially the same. Since one of the noise terms N_N^+ is the same for a conventional neuron implementation as for the ION, it appears that an ION implementation is not significantly different from a conventional neuron implementation in terms of immunity to noise and device imperfections for a given technology. Apparently the output is affected primarily by the variance of the noise and by the degree of spatial and temporal correlation, but not by the source of the noise. We conjecture that this result may not be peculiar to the ION model or to the particular net that we simulated, but is likely more general.

Two approaches have been described to implement the ION model; homogeneous (one device) and heterogeneous (two devices). A liquid crystal light valve response was given as an example for a homogeneous implementation. To demonstrate the feasibility of the ION, 2-D arrays of both analog and binary neuron units were experimentally demonstrated using an LCLV in a homogeneous implementation, successfully exhibiting both excitation and inhibition.

The authors would like to thank the faculty of the USC Center for the Integration of Optical Computing for numerous stimulating discussions, and the reviewers of this manuscript for helpful comments and suggestions. This work was supported by the Air Force Office of Scientific Research under grant AFOSR-86-0196 and University Research Initiative contract F49620-87-C-0007. Portions of this paper have been published and/or presented elsewhere.[40-43]

Appendix: Operational Analysis of the ION Model

A. Device Requirements

To guarantee the proper operation of the ION model, the input signals and the device characteristics must satisfy the following inequalities:

$$0 \le I_{\text{inh}} \le a_1, \tag{A1}$$

$$0 \le I_{\text{exc}} \le a_1, \tag{A2}$$

$$\alpha \ge a_1 \text{ (heterogeneous case)}, \tag{A3}$$

$$\alpha \ge a_1 + a_2 \text{ (homogeneous case)}. \tag{A4}$$

Let I_r be the maximum residual output of the device, i.e., the worst-case functional zero output, and a_2 and α are defined as the device inputs that correspond to an output of I_r (Fig. 4). Terms I_r, a_2, and α are chosen to provide an acceptable extinction ratio at the same time as sufficient separation between the I and N element active regions. Equation (A1) results from the limited domain of the I element. Equation (A3) comes from the requirement that the bias beam be non-negative. For the homogeneous case, we need a more strict constraint on α, to prevent the N element from operating in the inverting ($\bar{I}$) region. Since the smallest possible input to the N element is $I_{\text{in}}^{(N)} = I_{\text{bias}}$ (i.e., all other terms are zero), we need $I_{\text{bias}} = (\alpha - a_1) \geq a_2$, which gives Eq. (A4). In the case of the inhibitory signals being maximum, we require the total input to the N element to satisfy:

$$a_2 \leq I_{\text{exc}} + (\alpha - a_1) \leq \alpha. \quad \text{(A5)}$$

The upper limit in Eq. (A5) is not necessary for all neuron models; we include it here so that even with maximum excitatory input, sufficient inhibitory input can be generated to set the neuron output to the zero state. Combining Eq. (A5) with the above result that $\alpha \geq a_1 + a_2$ and with the non-negativity constraint on I_{exc} yields Eq. (A2). The lower bound on α for both cases is also dependent on the threshold requirement as shown below.

B. Threshold Implementation

We make the reasonable assumption that the threshold θ has the same maximum variation range as the input signals, i.e., $-a_1 \leq \theta \leq a_1$. Referring to Fig. 4, $I_{\text{bias}} = \alpha - a_1$ is the bias point for zero threshold. To achieve a positive threshold, it is shifted left by an amount θ to $I_{\text{bias}}^{(+)} = \alpha - a_1 - \theta$. For the heterogeneous case, $I_{\text{bias}}^{(+)}$ must be positive, i.e.,

$$\alpha \geq a_1 + \theta \text{ (heterogeneous case)} \quad \text{(A6)}$$

and θ is in the range $0 \leq \theta \leq a_1$.

To prevent the N element from operating in the inverting region for the homogeneous case (Fig. 4), the new bias point $I_{\text{bias}}^{(+)}$ must fulfill the more strict inequality $(\alpha - a_1) - \theta \geq a_2$, i.e.,

$$\alpha \geq a_1 + a_2 + \theta \text{ (homogeneous case)} \quad \text{(A7)}$$

and θ is in the same range.

To realize a negative threshold, the bias point is shifted right by an amount $-\theta$ to bias point $I_{\text{bias}}^{(-)}$. This relaxes the original constraint on α [Eqs. (A3) and (A4)], so no new constraints are needed. A time varying global theshold can be implemented by varying the bias beam. The device requirements [Eqs. (A6) and (A7)] then reflect the maximum positive threshold expected, $\theta_{\max}$. If no other limitations are expected on θ, then $\theta = a_1$ can be used.

C. Fan-in and Fan-out

We will assume binary neuron units during calculation of the maximum fan-in and fan-out of the ION. As shown in Fig. 3(c), the output of the jth neuron unit can be formulated as

$$I_{j(\text{out})} = I_r + \Delta I_s V_j, \quad \text{(A8)}$$

where $V_j \in \{0,1\}$ is the output state of the neuron unit j and I_r and ΔI_s are the residual output and output difference due to a state change of the N element, respectively. Let the fan-in and fan-out be N_{in} and N_{out}, respectively; then the effective input to neuron unit i from neuron unit j will be $w_{ij} \times I_{j(\text{out})}/N_{\text{out}}$. Therefore, the summed input to one neuron unit i is:

$$I_{i(\text{in})} = \frac{I_r}{N_{\text{out}}} \sum_{j=1}^{N_{\text{in}}} w_{ij} + \frac{\Delta I_s}{N_{\text{out}}} \sum_{j=1}^{N_{\text{in}}} w_{ij} V_j. \quad \text{(A9)}$$

The first term is a noise term due to the residual output of the optical devices and the second term is the desired signal term. Two different requirements on the relative sizes of these two terms are considered.

For networks with small to moderate fan-in, we require the signal term to be greater than the noise term. To estimate the fan-in, we take the mean of the above equation on both sides. Assuming the mean value of the weight to be $\overline{w_{ij}}$ and considering the worst case of only one input being active yield

$$\langle I_{j(\text{in})} \rangle = \frac{I_r}{N_{\text{out}}} N_{\text{in}} \overline{w_{ij}} + \frac{\Delta I_s}{N_{\text{out}}} \overline{w_{ij}} \cdot \quad \text{(A10)}$$

The fan-in can be calculated by setting the signal term greater than the noise term, so

$$N_{\text{in}} \leq \frac{\Delta I_s}{I_r}, \quad \text{(A11)}$$

which can be taken as a measure of the extinction ratio of the N element.

For a neural network with large fan-in, we require instead that a constant fraction β of the maximum signal term be greater than the noise term in Eq. (A9). In this case, the second term in Eq. (A10) is multiplied by βN_{in}, and this yields $1/\beta \leq \Delta I_s/I_r$. Thus, the extinction ratio of the device gives a lower bound on β but is independent of the fan-in. For example, in many networks a $1/\beta$ ranging from 10 to 100 may be sufficient for the optical neuron while the maximum fan-in may be 10^3–10^4.

The maximum fan-out can be calculated by considering the input to a neuron unit under maximum excitation. This, taking the mean again, yields

$$\langle I_{i(\text{in})}^{(\max)} \rangle = \frac{I_r}{N_{\text{out}}} \overline{w_{ij}} N_{\text{in}}^{(\text{exc})} + \frac{\Delta I_s}{N_{\text{out}}} \overline{w_{ij}} N_{\text{in}}^{(\text{exc})}, \quad \text{(A12)}$$

where $N_{\text{in}}^{(\text{exc})}$ is the fan-in to the excitatory site. The first term is assumed much smaller than the second and will be neglected hereafter. We require the total input to the N element on these conditions to be sufficient to drive the N element full on

$$a_1 + \gamma_s \frac{\Delta I_s}{N_{\text{out}}} N_{\text{in}}^{(\text{exc})} \overline{w_{ij}} + (\alpha - a_1) \geq \Delta a_N + \alpha, \quad \text{(A13)}$$

where the first and last terms on the left-hand side represent the input due to the I element output (under minimum inhibition) and the bias, respectively. Term γ_s is the optical system loss from the output of one neuron unit to the input of the next, and Δa_N is the differential input required to turn the N element full on (Fig. 4). Rearranging yields:

$$\frac{N_{\text{out}}}{N_{\text{in}}^{(\text{exc})}} \le \gamma_s \overline{w_{ij}} \frac{\Delta I_s}{\Delta a_N}. \tag{A14}$$

Thus, the ratio of the fan-out to the excitatory fan-in is bounded above by a measure of the differential gain of the N element, scaled by loss factors γ_s and $\overline{w_{ij}}$. During implementation of the ION model, with some optical devices ΔI_s can be controlled, within bounds, by the intensity of the readout beam; also, if necessary self-feedback can be employed on the N element. Both methods effectively increase the differential gain of the N element and permit a larger fan-out.

The residual output of the device may be less important for digital optical computing, because the fan-in is low and we can offset the holding power or bias point to overcome the residual output. But it is crucial in optical neural networks. This is because the residual term will be multiplied by the weights; as the network learns this will produce a time varying deterministic noise term in the summed input. If the noise is small and the weight modification varies sufficiently slowly, this noise term may not affect the short term dynamics of the neural network. Ideally, we need the residual term to be as small as possible.

We also note that the device requirements given in Eqs. (A1) and (A2) can now be rewritten in terms of known quantities as:

$$\Delta I_s \frac{N_{\text{in}}^{(\text{inh})}}{N_{\text{out}}} \le a_1,$$
$$\Delta I_s \frac{N_{\text{in}}^{(\text{exc})}}{N_{\text{out}}} \le a_1, \tag{A15}$$

where $N_{\text{in}}^{(\text{inh})}$ is the fan-in to the inhibitory site and the worst-case assumption of all weights being equal to one has been employed.

References

1. D. Z. Anderson and D. M. Lininger, "Dynamic Optical Interconnects: Volume Holograms as Optical Two-Port Operators," Appl. Opt. 26, 5031–5038 (1987).
2. J. F. Ebersole, "Optical Image Subtraction," Opt. Eng. 14, 436–447 (1975).
3. E. Marom and J. Grinberg, "Subtraction of Images with Incoherent Illumination in Real Time," Appl. Opt. 16, 3086–3087 (1977).
4. D. Psaltis, X.-G. Gu, and D. Brady, "Fractical Sampling Grids for Holographic Interconnections," Proc. Soc. Photo-Opt. Instrum. Eng. 963, 468–474 (1988).
5. D. Psaltis, D. Brady, and K. Wagner, "Adaptive Optical Networks Using Photorefractive Crystals," Appl. Opt. 27, 1752–1759 (1988).
6. N. H. Farhat, D. Psaltis, A. Prata, and E. Pack, "Optical Implementation of the Hopfield Model," Appl. Opt. 24, 1469–1475 (1985).
7. I. Shariv and A. A. Friesem, "All-Optical Neural Network with Inhibitory Neurons," Opt. Lett. 14, 485–487 (1989).
8. R. D. TeKolste and C. C. Guest, "Optical Competitive Neural Network with Optical Feedback," in *Proceedings, IEEE First International Conference on Neural Networks*, Vol. 3 (1987), pp. 625–629.
9. W. S. McCulloch and W. H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," Bull. Math. Biophys. 5, 115–133 (1943).
10. S. Grossberg, "Contour Enhancement, Short Term Memory, and Constancies in Reverberating Neural Networks," Studies in Appl. Math. 52, 213–257 (1973).
11. K. Fukushima, "Cognitron: a Self-Organizing Multilayered Neural Network," Biol. Cybernet. 20, 121–136 (1975).
12. S-I. Amari, "Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements," IEEE Trans. Comput. C-21, 1197–1206 (1972).
13. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Ability," Proc. Natl. Acad. Sci. U.S.A. 79, 2554–2558 (1982).
14. K. Fukushima, "Neocognitron: a Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," Biol. Cybernet. 36, 193–202 (1980).
15. S. Miyake and K. Fukushima, "A Neural Network Model for the Mechanism of Feature Extraction," Biol. Cybernet. 50, 377–384 (1984).
16. K. Fukushima, "A Hierarchical Neural Network Model for Associative Memory," Biol. Cybernet. 50, 105–113 (1984).
17. K. Fukushima, "A Neural Network Model for Selective Attention in Visual Pattern Recognition," Biol. Cybernet. 55, 5–15 (1986).
18. K. Fukushima, "Neural Network Model for Selective Attention in Visual Pattern Recognition and Associative Recall," Appl. Opt. 26, 4985–4992 (1987).
19. S. Grossberg, "On the Development of Feature Detectors in the Visual Cortex with Applications to Learning and Reaction-Diffusion Systems," Biol. Cybernet. 21, 145–159 (1976).
20. S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors," Biol. Cybernet. 23, 121–134 (1976).
21. S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: II. Feedback Expectation, Olfaction, Illusions," Biol. Cybernet. 23, 187–202 (1976).
22. G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," Comput. Vision Graphics Image Process. 37, 54–115 (1987).
23. G. A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," Appl. Opt. 26, 4919–4930 (1987).
24. C. von der Malsburg, "Self-Organization of Orientation Sensitive Cells in the Striate Cortex," Kybernetik 14, 85–100 (1973).
25. A. F. Gmitro and G. R. Gindi, "Optical Neurocomputer for Implementation of the Marr-Poggio Stereo Algorithm," in *Proceedings, IEEE First International Conference on Neural Networks*, Vol. 3 (1987), pp. 599–606.
26. S. A. Ellias and S. Grossberg, "Pattern Formation, Contrast Control and Oscillations in Short Term Memory of Shunting On-Center Off-Surround Networks," Biol. Cybernet. 20, 69–98 (1975).
27. M. Wang and A. Freeman, *Neural Function* (Little, Brown, Boston, 1987).
28. C. F. Stevens, "The Neuron," Sci. Am. 241, 54–65 (1979).
29. G. M. Shepherd, "Microcircuits in the Nervous System," Sci. Am. 238, 92–103 (1978).
30. A. C. Walker, "Application of Bistable Optical Logic Gate Arrays to All-Optical Digital Parallel Processing," Appl. Opt. 25, 1578–1585 (1986).
31. D. A. B. Miller *et al.*, "The Quantum Well Self-Electrooptic Effect Device: Optoelectronic Bistability and Oscillation, and Self-Linearized Modulation," IEEE J. Quantum Electron. QE-21, 1462–1476 (1985).
32. A. L. Lentine *et al.*, "Symmetric Self-Electro-Optic Effect Device: Optical Set-Reset Latch," Appl. Phys. Lett. 52, 1419–1421 (1988).
33. W. P. Bleha *et al.*, "Application of the Liquid Crystal Light

Valve to Real-Time Optical Data Processing," Opt. Eng. 17, 371–384 (1978).

34. B. K. Jenkins, A. A. Sawchuk, T. C. Strand, R. Forchheimer, and B. H. Soffer, "Sequential Optical Logic Implementation," Appl. Opt. 23, 3455–3464 (1984).

35. C. W. Stirk, S. M. Rovnyak, and R. A. Athale, "Effects of System Noise on an Optical Implementation of an Additive Lateral Inhibitory Network," in *Proceedings, IEEE First International Conference on Neural Networks*, Vol. 3 (1987), pp. 615–624.

36. B. K. Jenkins and C. H. Wang, "Model for an Incoherent Optical Neuron that Subtracts," Opt. Lett. 13, 892–894 (1988).

37. S. Grossberg, "Biological Competition: Decision Rules, Pattern Formation, and Oscillations," Proc. Natl. Acad. Sci. U.S.A. 77, 2338–2342 (1980).

38. S-I. Amari, "Competitive and Cooperative Aspects in Dynamics of Neural Excitation and Self-Organization," in *Proceedings, Competition and Cooperation in Neural Nets*, S. Amari and M. A. Arbib, Eds. (Springer-Verlag, New York, 1982), pp. 1–28.

39. U. Efron, E. Marom, and B. H. Soffer, "Array Division by Optical Computing," in *Topical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1985), paper TuF2.

40. B. K. Jenkins and C. H. Wang, "Model for an Incoherent Optical Neuron that Subtracts," J. Opt. Soc. Am. A 4(13), P127 (1987).

41. C. H. Wang and B. K. Jenkins, "Implementation Considerations of a Subtracting Incoherent Optical Neuron," in *Proceedings, IEEE First International Conference on Neural Networks*, Vol. II, (1988), pp. 403–410.

42. C. H. Wang and B. K. Jenkins, "Implementation of a Subtracting Incoherent Optical Neuron," in *Proceedings, IEEE Third Annual Parallel Processing Symposium*, Fullerton, CA (Mar. 1989).

43. C. H. Wang and B. K. Jenkins, "Subtracting Incoherent Optical Neuron: Experimental Demonstration," in *Technical Digest, OSA Annual Meeting* (Optical Society of America, Washington, DC, 1989), paper WU1.

Complexity Implications of Optical Parallel Computing

C. Lee Giles
Air Force Office of Scientific Research/NE
Bolling AFB, D.C. 20032-6448

and

B. Keith Jenkins
Signal and Image Processing Institute MC-0272
University of Southern California, Los Angeles, California 90089-0272

ABSTRACT

Abstract parallel computational models are discussed and related to optical computing. Two classes of parallel computing models, shared memory and graph/network, are used to analyze some of the possible effects of optical technology on parallel computing. It is found that the use of optics potentially provides certain fundamental advantages in communication and implementation of the architectures based on these models. In addition, some factors that limit the communication capabilities of optical systems for network models are discussed.

INTRODUCTION

In this paper we look at computational models for parallel processing in an attempt to increase our understanding of the role that optical computing might play. Most of the parallel architectures discussed in the parallel processing community are heavily influenced by the constraints of electronic systems. The purpose of our approach in this paper is to abstract the notion of parallel computing from the limitations of any given technology. This abstract model can then be used as a starting point for the design of parallel optical computing architectures. In the process, some of the consequences of inherent differences between optical and electronic systems start to become apparent.

Computing paradigms are important for understanding the level and class of problems that the computer scientist is addressing. Consider the following structural paradigmatic classification: physical, functional, computational. A representation and example of each of these paradigms is illustrated in Table 1. Here we are only concerned with the computational paradigm and the optical implications.

Before discussing computational models, both sequential and parallel, we define computational order or complexity as it is used in this paper. For a problem of size n, the order of its time (or space) complexity is denoted by $O(f(n))$, which is defined such that $f(n)$ is the asymptotic behavior of that problem as n grows very large. $O(f(n))$ represents an asymptotic upper bound, and $\Omega(g(n))$, defined similarly, represents an asymptotic lower bound. This notation will be used throughout the paper. For formal definitions see (Gottlieb and Kruskal, 1984).

Computational models are important because they measure the performance of general classes of both sequential and parallel algorithms on an idealized abstract machine. However, the performance of these models is highly dependent on the class of algorithms. If the generic class of algorithms is known for a specific problem (e.g. the communication algorithms of broadcasting, reporting, sorting, etc.), then the computational model which efficiently runs these algorithms would be a starting point for the design of a computer architecture that would do the same. The basic assumption is that algorithms which run well on a computational model should run well on the model-derived architecture. Our intention is to show that optics has a greater potential than electronics for physically realizing key aspects of some of these computational models.

SEQUENTIAL COMPUTATIONAL MODELS

Since parallel computational models are for the most part extensions of sequential models, we briefly discuss these sequential machines with particular emphasis on the Random Access Machine. The most well-known and basic of the sequential machines is the Turing machine (TM). The universal TM has the capability of computing any algorithm that is computable (a rather circular thesis since a universal

Table 1. Processing paradigm levels.

PARADIGMS	REPRESENTATION	EXAMPLE
Physical	Hardware/Technology	IC, Board
Functional	Architecture	PE, Memory, Interconnection Topology
Computational	Algorithms/Metrics	Turing Machine, Automata, Random Access Machine

TM defines what is computable). A principal application of the TM is in determining lower bounds on the space or time necessary to solve algorithmic problems. The TM is a well-known computational model; for further interest see the very informative text by Minsky (1967).

The Random Access Machine (RAM) (Aho, Hopcroft, and Ullman, 1974) is a less primitive computational model which can be stylized as a primitive computer. The RAM model is a one-accumulator computer in which the instructions are not allowed to modify themselves. A RAM consists of a read-only input tape, a write-only output tape, a program and a memory. The time on the RAM is bounded above by a polynomial function of time on the TM. In particular, for a TM of time complexity $T(n) \geq n$, a RAM can simulate the TM in $O(T(n))$ or $O(T(n)\log n)$ time, depending on the cost function used for the RAM. For the converse, using a TM to simulate a RAM, the bounds on time required by the TM are higher and are highly dependent on the RAM cost function used (Aho, Hopcroft, and Ullman, 1974). The program of a RAM is not stored in memory and is unmodifiable. The RAM instruction set is is small and consists of operations such as store, add, subtract, and jump if greater than zero; indirect addresses are permitted. A common RAM model is the uniform cost one, which assumes that each RAM instruction requires one unit of time and each register one unit of space. Attempts to parallelize the RAM computational model resulted in many of the parallel computational models.

SHARED MEMORY MODELS

We will discuss only two classes of parallel computational models; shared-memory models and graph/network models. As might be inferred from the shared memory term, these models are based on global memories and are differentiated by their accessibility to memory. In Fig. 1 we see a typical shared memory model where individual processing elements (PE's) have variable simultaneous access to an individual memory cell. (A processing element is a physically isolated computational unit consisting of some local memory and computational power. A PE can be construed as a computational primitive from which more sophisticated architectures can be constructed (Hwang and Briggs, 1984)). Each PE can access any cell of the global memory in unit time. In addition, many PE's can access many different cells of the global memory simultaneously. In the models we discuss, each PE is a slightly modified RAM without the input and output tapes, and with a modified instruction set to permit access to the global memory. A separate input for the machine is provided. A given processor can generally not access the local memory of other processors.

The models differ primarily in whether they allow simultaneous reads and/or writes to the *same* memory cell. The PRAC, parallel random access computer (Lev, Pippenger and Valiant, 1981) does not allow simultaneous reading or writing to an individual memory cell. The PRAM, parallel random access machine, (Fortune and Wyllie, 1978) permits simultaneous reads but not simultaneous writes to an individual memory cell. The WRAM, parallel write random access machine, denotes a variety of models that permit simultaneous reads and certain writes, but differ in how the write conflicts are resolved. For example, a model by Shiloach and Vishkin (1981) allows a simultaneous write only if all processors are trying to write the same value. The paracomputer (Schwartz, 1980) has simultaneous writes but only "some" of all the information written to the cell is recorded. The models represent a hierarchy of time complexity given by

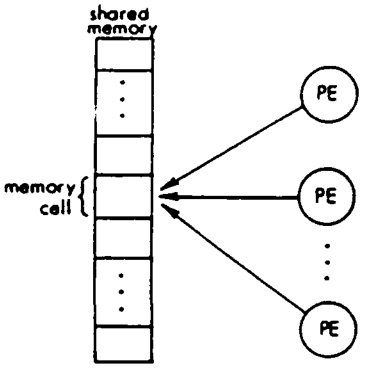


Fig. 1. Conceptual diagram of shared memory models.

$$T^{\mathrm{PRAC}} \geq T^{\mathrm{PRAM}} \geq T^{\mathrm{WRAM}}$$

where T is the minimum number of parallel time steps required to execute an algorithm on each model. More detailed comparisons are dependent on the algorithm (Borodin and Hopcroft, 1985).

Implications of optics

In general, none of these shared memory are physically realizable because of actual fan-in limitations. Optical interconnections permit greater fan-in than electronic systems. In addition, the non-interacting property of photons in a linear medium (versus the mutual interaction of electrons) may permit simultaneous memory reads much more easily. As an electronic example, the ultracomputer (Schwartz, 1980) is an architectural manifestation of the paracomputer that uses a hardwired Omega network between the PE's and memories; it simulates the paracomputer within a time penalty of $O(\log^2 n)$.

Optical systems could in principle be used to implement this parallel memory read capability. As a simple example, a single 1-bit memory cell can be represented by one pixel of a 1-D or 2-D array; the bit could be represented by the state (opaque or transparent) of the memory cell. Many optical beams can simultaneously read the contents of this memory cell without contention (Fig. 2). In addition to this an interconnection network is needed between the PE's and the memory, that can allow any PE to communicate with any memory cell, preferably in one step, and with no contention. A regular crossbar is not sufficient for this because fan-in to a given memory cell must be allowed. Figure 3 shows a conceptual block diagram of a system based on the PRAM model; here the memory array operates in reflection instead of transmission. The fan-in required of the interconnection network is also depicted in the figure.

Optical systems can potentially implement crossbars that also allow this fan-in. Several optical crossbar designs discussed in (Sawchuk, et al., 1986) exhibit fan-in capability. An example is the optical crossbar shown schematically in Fig. 4. The 1-D array on the left could be optical sources (LED's or

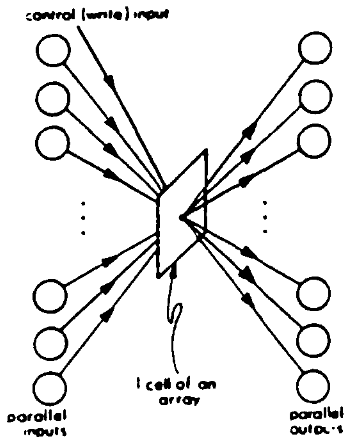


Fig. 2. One memory cell of an array, showing multiple optical beams providing contention-free read access.

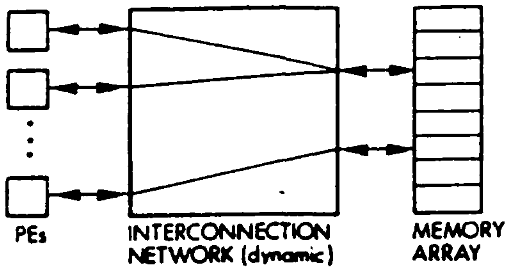


Fig. 3. Block diagram of an optical architecture based on parallel RAM models.

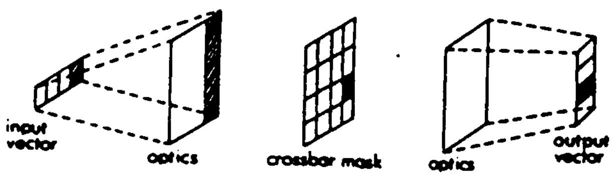


Fig. 4. Example of an optical crossbar interconnection network.

laser diodes) or just the location of optical signals entering from previous components. An optical system spreads the light from each input source into a vertical column that illuminates the crossbar mask. Following the crossbar mask, a set of optics collects the light transmitted by each row of the mask onto one element of the output array. The states of the pixels in the crossbar mask (transparent or opaque) determine the state of the crossbar switch. Multiple transparent pixels in a column provide fanout; multiple transparent pixels in a row provide fan-in. Many optical reconfigurable network designs are possible, and provide tradeoffs in performance parameters such as bandwidth, reconfiguration time, maximum number of lines, hardware requirements, etc. Unfortunately, most simple optical crossbars will be limited in size to approximately 256 x 256 (Sawchuk, et al., 1986). We are currently considering variants of this technique to increase the number of elements. Possibilities include using a multistage but nonblocking interconnection network (e.g. Clos), a hierarchy of crossbars, and/or a memory hierarchy.

GRAPH/NETWORK MODELS

Graph/network models are characterized by a collection of usually identical PE's that are interconnected with a fixed network. They can be represented by graphs, with a node of the graph for each PE and an arc or link of the graph for each PE to PE interconnection. The models differ from one another in the length of time required for a message to traverse one arc of the graph, and on the assumptions placed on the PE's such as their ability to respond to multiple messages. The feasibility of implementation of these models depends on the connectivity of the graph; if the connectivity is not too high, the model is much more readily implemented than the shared memory models.

Network models can be compared to shared memory models. Any of the shared memory models can efficiently simulate (in $O(1)$ time) a network model. This is done by dedicating a different cell of the global memory for each link of the network. One PE sends a message to another by writing the message to a memory cell which the other PE then reads. Conversely, suppose the network model is capable of (partial) routing in $r(n)$ time. Then it can simulate *one* step of the PRAC, PRAM, or WRAM in $O(r(n))$ time (Borodin and Hopcroft, 1985).

In a highly parallel machine communications are exceedingly important and for many tasks can dominate the execution time of the algorithm. We therefore concentrate on communications in our analysis of these models. We will use communication algorithms for our analysis since they have been found to be reasonable predictors of performance (Levitan, 1985).

PE Complexity and Communications

Since the performance of network models depends on the assumptions on the individual processing elements, we need to consider these assumptions and their relationships to communication tasks. We will show that in general the communications between PE's (or the network topology) cannot be completely decoupled from the hardware complexity of the PE's themselves. After giving a relationship between PE space complexity and interconnection capability, we will be able to identify what reasonable assumptions on the PE complexity are for the optics and electronics cases. These assumptions will be used in assessing the performance of different communication tasks on network models. In this paper the term PE complexity refers to the *space* complexity of *each* PE. We will not discuss time complexity of individual PE's.

For simplicity, we will assume the bandwidth of each I/O line to a PE is fixed and is given. Thus we are *a priori* not considering one of the potential advantages of an optical system over an electronic one. We will, however, consider the effect of the number of I/O lines to a PE. The complexity of each PE must grow at least as fast as the number of I/O lines to the PE. Thus a lower bound on the PE complexity is $\Omega(d)$, where d is the number of I/O lines to the PE. This lower bound applies even in the simplest case, in which all input lines are combined (e.g. by a logic operation), and the output lines are not allowed to carry distinct signals in a single time step. If signals on different input lines must be kept distinct yet are accepted in a single time step, or if distinct signals can be put on multiple output lines in a single step then the PE time or space complexity will be higher.

Implications of this lie in the communication ability of PE networks, particularly in the optics case. With electronic technology, the number of I/O lines to a PE is generally quite limited and this limits the ability of the PE's to communicate. This is due to limited pinout, cost of interconnections, etc. The PE complexity is in practice not an issue for communications. In the optics case, however, there are no pinout restrictions and many parallel interconnection lines are feasible. However, there *are* limitations on the total number of interconnections in an optical PE network; these are due to the PE complexity itself. In other words, the PE's have to be able to accommodate all of the I/O lines. The optics case apparently allows a balance between the interconnections and the PE complexity; in the electronics case the interconnections are further limited by technology factors.

Communication Time and Space of Network Models

In this section we will give the time required to execute different communication tasks on different network topologies, and values of some measures of space complexity required in implementing different network topologies in VLSI and optics. We are concerned with fine-grained systems, that is systems with a large or very large number of relatively simple PE's. As a minimum, we assume each PE can store its own address so that it knows where it is located. Many algorithms can become quite difficult without this feature. This implies that the PE complexity must be allowed to grow $\Omega(\log N)$.

In an electronic system, the number and length of interconnections is important and ideally should be minimized. The number of connections to each PE or node of the graph is limited to small values due to I/O constraints. This limits the connectivity of graphs that can be efficiently implemented. The *degree* of a graph is the number lines connected to each node. Electronic systems limit the degree of the graph to a relatively small value; for large enough N the degree must be a constant, independent of N.

Optical systems have no I/O restrictions on the PE's *per se*, but as discussed above the degree of the graph will be limited by the complexity of the PE's. Since the PE complexity is $\Omega(\log N)$ anyway, in the optics case the degree of the graph can easily be $O(\log N)$. Larger degrees, e.g. $O(N^{1/p})$, where $p \geq 2$ may also be feasible.

In order to calculate communication times on a network model, certain assumptions need to be specified. We assume that all messages are the same size and are routed to their destinations over the fixed connection network by passing over links and through PE's. One time step is defined as the time for a PE to send a message, the message to travel over one link, be received by the PE at the end of the link, and for the PE to perform any computation on the message (such as altering its tag or combining messages that arrive simultaneously). The processors operate synchronously. Finally, the number of messages that can simultaneously be accepted or output by each PE must be considered. In the electronics case, the number of messages that can be simultaneously accepted by a PE is relatively small (because of the degree limitation), and will probably need to be a constant independent of N (Kushner and Rosenfeld, 1983). For simplicity this can be taken to be 1. A PE can output identical copies of the same message, but not multiple messages. For the optics case, we assume only a limit on the PE complexity; this then dictates how flexible the inputs and outputs of the PE can be. We limit the PE complexity to the degree of the network or $\log N$, whichever is greater. Each PE can accept d simultaneous messages, where d is the degree of the network, and may increase with N. Each PE can output d identical messages simultaneously; outputing different messages simultaneously (in conjunction with inputing several messages simultaneously) can involve an increase in PE complexity, depending on what the PE is required to do.

Kushner and Rosenfeld (1983) classify communication tasks as one-to-many, many-to-one, and one-to-one. One to many tasks include broadcasting, in which one PE (the root if there is a node so distinguished) sends the same message to many (or all) other PE's; in general the messages may be altered as they travel to their destinations. Many to one tasks must be divided into two classes. In both classes many PE's all send messages to the same PE (root). In one case, *condensing*, the messages can be combined (e.g. added) in route to the destination; in the case of *funneling*, the messages must be kept separate. One to one tasks are permutations, in which each PE sends a message to one other PE. In the worst case, half the PE's send messages to the other half, each message with a different destination PE.

Optical fixed interconnections can be implemented using lenses, free space propagation, and computer-generated holograms. Let N be the number of nodes being interconnected. In an optical holographic interconnection system, a 2-D array of nodes is connected to a 2-D array of nodes. A set I of interconnection patterns can be defined; each interconnection pattern can be represented by an ordered pair representing the location or address of the destination of the destination node in the array relative to the location of the source node (multiple ordered pairs can be used to accommodate fanout). In one scenario, all interconnections that are implemented must be in I. Let M be the number of distinct interconnection patterns that are used. The primary limiting factor on the number and types of interconnections that can be implemented in the optics case is given by an upper limit on the product MN (Jenkins, et al., 1984). This is proportional to the number of resolvable elements that the hologram(s) must contain. Current hologram plotting devices give the approximate limit $MN \leq 4\times10^8$. MN is a measure of the space complexity or hologram area of interconnections implemented with this optical technique. Figure 5 shows an optical system

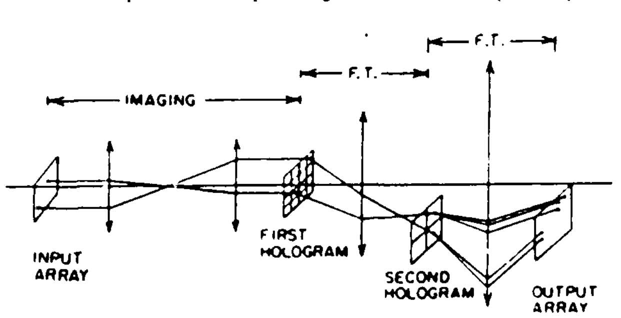


Fig. 5. Optical system for implementing fixed interconnections.

that can implement fixed interconnections; it connects a 2-D array to a 2-D array. The holograms provide the beam splitting (fanout) and beam steering operations. Each input node is transferred to a corresponding pixel of the first hologram

Table 2.

Network Topology	Degree	Complexity of each PE	Time Complexity			Space Complexity	
			Broadcasting & Condensing	Permuting	Funneling	VLSI Area	Hologram Complexity
Array (nearest neighbor)	4	logN	$\sqrt{N}$ (4)	N? (4)	N (4)	N	N
Pyramid	5	logN	logN (4)	N (4)	N (4)	?	N log N
Hypercube	log N	logN	logN (4)	logN (4)	N/logN (3)	N^2	N log N
Tree	$1+b \approx N^{1/p}$	$N^{1/p}$	$\log_b N = p$	$N^{1-1/p}$ (2)	$N^{1-1/p}$ (3)	?	N
Fully Connected	N	N	1 (4)	1 (4)	1 (3)	N^3-N^4	N^2

(1) b = branching factor of tree = $m^{1/p}$, where m = no. of leaf nodes. p = radius = no. of levels -1. p≥1.

(2) Allowing the root node to have complexity $O(N^{2/p})$.

(3) Depending on algorithm requirements, could require more time or higher space complexity of the destination (root) PE.

(4) *From Kushner and Rosenfeld (1983).*

The second hologram stores the different interconnection patterns, and the first hologram addresses the appropriate interconnection pattern(s) for each node. Each stored interconnection pattern can be used by multiple nodes simultaneously. This system can be used to implement any of the network topologies discussed in this section.

The worst case order of magnitude communication time for several networks of different topologies and degrees is given in Table 2. The array and binary tree take the same time under our optics assumptions as under our electronics assumptions. The optics assumptions allow degrees that are a function of N, which further reduce the time of communication tasks. The fully interconnected array is impractical for large N in both optical and electronic cases; however, the upper limit on N that is feasible is probably higher in the optics case than in electronics. Table 2 also gives asymptotic upper bounds for the optical interconnection space complexity (MN), and asymptotic lower bounds for area in VLSI. A common grid model was used for the VLSI area complexity (Ullman, J. D., 1984). While optical and electronic technologies are different, for large N differences in complexity can outweigh differences in technology.

CONCLUSIONS

Parallel computational models offer a fundamental basis for understanding parallel computation and abstract out the limitations posed by a particular technology. Actual parallel computer architectures designed with these models in mind should exhibit performance similar to the parallel computational models. We have found some potential advantages of implementing these parallel models in optical computing architectures; advantages such as better contention-free access to global memories, associated reconfigurable interconnection networks, and implementation of increased-degree PE networks. We also pointed out that the connectivity of optical systems is not unlimited, but limited by the complexity of the components that are being connected. Finally, based on these factors, we conjecture that optics provides the potential for allowing a closer realization of linear computational speed-up than electronics.

REFERENCES

Aho, A.V., J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, Mass. Addison-Wesley, 1974.

Borodin, A. and J.E. Hopcroft, "Routing, Merging, and Sorting on Parallel Models of Computation," *Journal of Computer and System Sciences*, Vol. 30, pp. 130-145 1985.

Fortune, S., and J. Wyllie, "Parallelism in Random Access Machines," *Proc 10th Annual ACM STOC*, San Diego, California, pp. 114-118, 1978.

Gottlieb, A., and C.P. Kruskal, "Complexity Results for Permuting Data and Other Computations on Parallel Processors," *J. ACM*, Vol. 31, No. 2, pp. 193-209, 1984.

Hopcroft, J.E. and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Reading, Mass., Addison-Wesley, 1979.

Hwang, K. and F. A. Briggs, *Computer Architecture and Parallel Processing*, New York, McGraw-Hill, 1984.

Jenkins, B.K., et al., "Architectural Implications of a Digital Optical Processor," *Applied Optics*, Vol. 23, No. 19, pp. 3465-3474, 1984.

Kushner, T. R. and A. Rosenfeld, "Interprocessor Communication Requirements for Parallel Image Processing," *Proc. IEEE 1983 Computer Architecture for Pattern Analysis and Image Database Management*, IEEE Cat. No. 83CH1929-9, pp. 177-183, 1983.

Lev, G., N. Pippenger, and L.G. Valiant, "A Fast Parallel Algorithm for Routing in Permutation Networks" *IEEE Trans. on Computers*, Vol. C-30, No. 2, pp. 33-100, Feb 1981.

Levitan, S.P., "Evaluation Criteria for Communication Structures in Parallel Architectures," Proc. 1985 International Conference on Parallel Processing, IEEE Cat. No. 85CH2140-2, p. 147, 1985.

Minsky, M. L., *Computation: Finite and Infinite Machines*, Engelwood Cliffs, N. J., Prentice-Hall, 1967.

Sawchuk, A. A., B.K. Jenkins, C.S. Raghavendra, and A. Varma, "Optical Matrix-Vector Implementations of Crossbar Interconnection Networks," *Proc. International Conference on Parallel Processing*, St. Charles, IL, August 1986; also Sawchuk, et al., "Optical Interconnection Networks," *Proc. International Conference on Parallel Processing*, pp. 388-392, August 1985.

Schwartz, J.T., "Ultracomputers," *A.C.M. Trans on Prog. Lang. and Sys.*, Vol. 2, No. 4, pp. 484-521, October 1980.

Shiloach, Y., and Vishkin, U., "Finding the Maximum, Merging and Sorting in a Parallel Computation Model," *J. Algorithms*, pp. 88-102, March 1981.

Uhr, L. "Augmenting Pyramids and Arrays by Compounding them with Networks," *Proc. IEEE 1983 Computer Architecture for Pattern Analysis and Image Database Management*, IEEE Cat. No. 83CH1929-9, pp. 162-168, 1983.

Ullman, J.D., *Computational Aspects of VLSI*, Rockville, Maryland, Computer Science Press, 1984.

T-AS/38/6//34989

Stochastic and Deterministic Networks for Texture Segmentation

B. S. Manjunath
Tal Simchony
Rama Chellappa

Reprinted from
IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING
Vol. 38, No. 6, June 1990

Stochastic and Deterministic Networks for Texture Segmentation

B. S. MANJUNATH, STUDENT MEMBER, IEEE, TAL SIMCHONY, MEMBER, IEEE, AND RAMA CHELLAPPA, SENIOR MEMBER, IEEE

Abstract—This paper describes several texture segmentation algorithms based on deterministic and stochastic relaxation principles, and their implementation on parallel networks. The segmentation problem is posed as an optimization problem and two different optimality criteria are considered. The first criterion involves maximizing the posterior distribution of the intensity field given the label field (maximum *a posteriori* (MAP) estimate). The posterior distribution of the texture labels is derived by modeling the textures as Gauss Markov random field (GMRF) and characterizing the distribution of different texture labels by a discrete multilevel Markov model. Fast approximate solutions for MAP are obtained using deterministic relaxation techniques implemented on a Hopfield neural network and are compared with those of simulated annealing in obtaining the MAP estimate. A stochastic algorithm which introduces learning into the iterations of the Hopfield network is proposed. This iterated hill-climbing algorithm combines fast convergence of deterministic relaxation with the sustained exploration of the stochastic algorithms, but is guaranteed to find only a local minimum. The second optimality criterion requires minimizing the expected percentage of misclassification per pixel by maximizing the posterior marginal distribution, and the maximum posterior marginal (MPM) algorithm is used to obtain the corresponding solution. All these methods implemented on parallel networks can be easily extended for hierarchical segmentation and we present results of the various schemes in classifying some real textured images.

I. Introduction

THIS PAPER describes several algorithms, both deterministic and stochastic, for the segmentation of textured images. Segmentation of image data is an important problem in computer vision, remote sensing, and image analysis. Most objects in the real world have textured surfaces. Segmentation based on texture information is possible even if there are no apparent intensity edges between the different regions. There are many existing methods for texture segmentation and classification, based on different types of statistics that can be obtained from the gray level images. The approach we use stems from the idea of using Markov random field models for textures in an image. We assign two random variables for the observed pixel, one characterizing the underlying intensity and the other for labeling the texture corresponding to the pixel location. We use the Gauss Markov random field (GMRF) model for the conditional density of the intensity field given the label field. Prior information about the texture label field is introduced using a discrete Markov distribution. The segmentation can then be formulated as an optimization problem involving minimization of a Gibbs energy function. Exhaustive search for the optimum solution is not possible because of the large dimensionality of the search space. For example, even for the very simple case of segmenting a 128×128 image into two classes, there are $2^{2^{14}}$ possible label configurations. Derin and Elliott [1] have investigated the use of dynamic programming for obtaining an approximation to the maximum *a posteriori* (MAP) estimate while Cohen and Cooper [2] give a deterministic relaxation algorithm for the same problem. The optimal MAP solution can be obtained by using stochastic relaxation algorithms such as simulated annealing [3]. Recently there has been considerable interest in using neural networks for solving computationally hard problems and the main emphasis in this paper is on developing parallel algorithms which can be implemented on such networks of simple processing elements (neurons).

The inherent parallelism of neural networks provides an interesting architecture for implementing many computer vision algorithms [4]. Some examples are image restoration [5], stereopsis [6], and computing optical flow [7]–[9]. Networks for solving combinatorially hard problems such as the traveling salesman problem have received much attention in the neural network literature [10]. In almost all cases, these networks are designed to minimize an energy function defined by the network architecture. The parameters of the network are obtained in terms of the energy (cost) function it is designed to minimize and it can be shown [10] that for networks having symmetric interconnections, the equilibrium states correspond to the local minima of the energy function. For practical purposes, networks with few interconnections are preferred because of the large number of processing units required in any image processing application. In this context Markov random field (MRF) models for images play a useful role. They are typically characterized by local dependencies and symmetric interconnections which can be expressed in terms of energy functions using Gibbs-Markov equivalence [3].

Manuscript received September 19, 1988; revised August 2, 1989. This work was supported in part by the AFSOR under Grant 86-0196.

B. S. Manjunath and R. Chellappa are with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089.

T. Simchony was with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089. He is now with ECI Telecom, Rafael, Israel

IEEE Log Number 9034989

We look into two different optimality criteria for segmenting the image. The first corresponds to the label configuration which maximizes the posterior probability of the label array given the intensity array. As noted before, an exhaustive search for the optimal solution is practically impossible. An alternative is to use stochastic relaxation algorithms such as simulated annealing [3], which asymptotically converge to the optimal solution. However the computational burden involved because of the theoretical requirements on the initial temperature and the impractical cooling schedules overweigh their advantages in many cases. Fast approximate solutions can be obtained by such deterministic relaxation algoritms as the iterated conditional mode rule [11]. The energy function corresponding to this optimality criterion can be mapped into a Hopfield-type network in a straightforward manner and it can be shown that the network converges to an equilibrium state, which in general will be a local optimum. The solutions obtained using this method are sensitive to the initial configurations, and in many cases starting with a maximum likelihood estimate is preferred. Stochastic learning can be easily introduced into the network, and the overall system improves the performance by learning while searching. The learning algorithms used are derived from the theory of stochastic learning automata [12] and we believe that this is the first time such a hybrid system has been used in an optimization problem. The stochastic nature of the system helps in preventing the algorithm from being trapped in a local minimum and we observe that this improves the quality of the solutions obtained.

The second optimality criterion minimizes the expected percentage of classification error per pixel. This is equivalent to finding the pixel labels that maximize the marginal posterior probability given the intensity data [13]. Since calculating the marginal posterior probability is very difficult, Marroquin [14] suggested the MPM algorithm, which asymptotically computes the posterior marginal. In [14] the algorithm is used for image restoration, stereo matching, and surface interpolation. Here we use this method to find the texture label that maximizes the marginal posterior probability for each pixel.

The organization of the paper is as follows: Section II describes the image model. A neural network model for the relaxation algorithms is given in Section III along with a deterministic updating rule. Section IV discusses the stochastic algorithms for segmentation and their parallel implementation on the network. A learning algorithm is proposed in Section V and the experimental results are provided in Section VI.

II. Image Model

The use of MRF models for image processing applications has been investigated by many researchers (see e.g., Chellappa [15]). Cross and Jain [16] provide a detailed discussion on the application of MRF in modeling textured images. Geman and Geman [3] discuss the equivalence between MRF and Gibbs distributions. The GMRF model for the texture intensity process has been used in

		7	6	7		
	5	4	3	4	5	
7	4	2	1	2	4	7
6	3'	1	x	1	3	6
7	4	2	1	2	4	7
	5	4	3	4	5	
		7	6	7		

Fig. 1. Structure of the GMRF model. The numbers indicate the order of the model relative to x [16].

[1], [2], and [17]. The MRF is also used to describe the label process in [1] and [2]. In this paper we use the fourth-order GMRF indicated in Fig. 1 to model the conditional probability density of the image intensity array given its texture labels. The texture labels are assumed to obey a first- or second-order discrete Markov model with a single parameter β, which measures the amount of clustering between adjacent pixels.

Let Ω denote the set of grid points in the $M \times M$ lattice, i.e., $\Omega = \{(i, j), 1 \leq i, j \leq M\}$. Following Geman and Graffigne [18], we construct a composite model which accounts for texture labels and gray levels. Let $\{L_s, s \in \Omega\}$ and $\{Y_s, s \in \Omega\}$ denote the labels and zero mean gray level arrays respectively. The zero mean array is obtained by subtracting the local mean computed in a small window centered at each pixel. Let N_s denote the symmetric fourth-order neighborhood of a site s. Then, assuming that all the neighbors of s also have the same label as that of s, we can write the following expression for the conditional density of the intensity at the pixel site s:

$$P(Y_s = y_s \mid Y_r = y_r, r \in N_s, L_s = l) = \frac{\exp\left[-U(Y_s = y_s \mid Y_r = y_r, r \in N_s, L_s = l)\right]}{Z(l \mid y_r, r \in N_s)} \qquad (1)$$

where $Z(l \mid y_r, r \in N_s)$ is the partition function of the conditional Gibbs distribution and

$$U(Y_s = y_s \mid Y_r = y_r, r \in N_s, L_s = l) = \frac{1}{2\sigma_l^2}\left(y_s^2 - 2\sum_{r \in N_s} \Theta_{s,r}^l y_s y_r\right). \qquad (2)$$

In (2), σ_l and Θ^l are the GMRF model parameters of the lth texture class. The model parameters satisfy $\Theta_{r,s}^l = \Theta_{-r,s}^l = \Theta_{s-r}^l = \Theta_r^l$.

We view the image intensity array as composed of a set of overlapping $k \times k$ windows W_s, centered at each pixel $s \in \Omega$. In each of these windows we assume that the texture label L_s is homogeneous (all the pixels in the window belonging to the same texture) and compute the joint distribution of the intensity in the window conditioned on L_s. The corresponding Gibbs energy is used in the relaxation process for segmentation. As explained in the previous paragraph, the image intensity in the window is modeled by a fourth-order stationary GMRF. The local mean is computed by taking the average of the intensities in the window W_s and is subtracted from the original image to get the zero mean image. All our references to the intensity array correspond to the zero mean image. Let Y_s^* denote the 2-D vector representing the intensity array in the window W_s. Using the Gibbs formulation and assuming a free boundary model, the joint probability density in the window W_s can be written as [2]

$$P(Y_s^* \mid L_s = l) = \frac{\exp\left[-U_1(Y_s^* \mid L_s = l)\right]}{Z_1(l)}$$

where $Z_1(l)$ is the partition function and

$$U_1(Y_s^* \mid L_s = l) = \frac{1}{2\sigma_l^2} \sum_{r \in W_s} \left\{ y_r^2 - \sum_{\tau \in N^* \mid r+\tau \in W_s} \cdot\, \Theta_\tau^l y_r (y_{r+\tau} + y_{r-\tau}) \right\}. \tag{3}$$

N^* is the set of shift vectors corresponding to a fourth-order neighborhood system:

$$\begin{aligned} N^* &= \{\tau_1, \tau_2, \tau_3, \cdots, \tau_{10}\} \\ &= \{(0, 1), (1, 0), (1, 1), (-1, 1), (0, 2), (2, 0), \\ &\quad (1, 2), (2, 1), (-1, 2), (-2, 1)\}. \end{aligned}$$

The label field is modeled as a first- or second-order discrete MRF. If $\hat{N}_s$ denotes the appropriate neighborhood for the label field, then we can write the distribution function for the texture label at site s conditioned on the labels of the neighboring sites as

$$P(L_s \mid L_r, r \in \hat{N}_s) = \frac{e^{-U_2(L_s \mid L_r)}}{Z_2}$$

where Z_2 is a normalizing constant and

$$U_2(L_s \mid L_r, r \in \hat{N}_s) = -\beta \sum_{r \in \hat{N}_s} \delta(L_s - L_r), \qquad \beta > 0. \tag{4}$$

In (4), β determines the degree of clustering, and $\delta(i - j)$ is the Kronecker delta. Using the Bayes rule, we can write

$$P(L_s \mid Y_s^*, L_r, r \in \hat{N}_s) = \frac{P(Y_s^* \mid L_s)\, P(L_s \mid L_r, r \in \hat{N}_s)}{P(Y_s^*)}. \tag{5}$$

Since Y_s^* is known, the denominator in (5) is just a constant. The numerator is a product of two exponential functions and can be expressed as

$$P(L_s \mid Y_s^*, L_r, r \in \hat{N}_s) = \frac{1}{Z_p} \exp\left[-U_p(L_s \mid Y_s^*, L_r, r \in \hat{N}_s)\right] \tag{6}$$

where Z_p is the partition function and $U_p(\cdot)$ is the posterior energy corresponding to (5). From (3) and (4) we write

$$U_p(L_s \mid Y_s^*, L_r, r \in \hat{N}_s) = w(L_s) + U_1(Y_s^* \mid L_s) + U_2(L_s \mid L_r, r \in \hat{N}_s). \tag{7}$$

Note that the second term in (7) relates the observed pixel intensities to the texture labels and the last term specifies the label distribution. The bias term $w(L_s) = \log Z_1(L_s)$ is dependent on the texture class and it can be explicitly evaluated for the GMRF model considered here using the toroidal assumption (the computations become very cumbersome if toroidal assumptions are not made). An alternative approach is to estimate the bias from the histogram of the data as suggested by Geman and Graffigne [18]. Finally, the posterior distribution of the texture labels for the entire image given the intensity array is

$$P(L \mid Y^*) = \frac{P(Y^* \mid L)\, P(L)}{P(Y^*)}. \tag{8}$$

Maximizing (8) gives the optimal Bayesian estimate. Though it is possible in principle to compute the right-hand side of (8) and find the global optimum, the computational burden involved is so enormous that it is practically impossible to do so. However, we note that the stochastic relaxation algorithms discussed in Section IV require only the computation of (6) to obtain the optimal solution. The deterministic relaxation algorithm given in the next section also uses these values, but in this case the solution is only an approximation to the MAP estimate.

III. A Neural Network for Texture Classification

We describe the network architecture used for segmentation and the implementation of deterministic relaxation algorithms. The energy function which the network minimizes is obtained from the image model discussed in the previous section. For convenience of notation let $U_1(i, j, l) = U_1(Y_s^*, L_s = l) + w(l)$, where $s = (i, j)$ denotes a pixel site and $U_1(\cdot)$ and $w(l)$ are as defined in (7). The network consists of K layers, each layer arranged as an $M \times M$ array, where K is the number of texture classes in the image and M is the dimension of the image. The elements (neurons) in the network are assumed to be binary and are indexed by (i, j, l) where $(i, j) = s$ refers to their position in the image and l refers to the layer. The (i, j, l)th neuron is said to be ON if its output V_{ijl} is 1, indicating that the corresponding site $s = (i, j)$ in the image has the texture label l. Let $T_{ijl;i'j'l'}$ be the connection strength between neurons (i, j, l) and (i', j', l') and I_{ijl} be the input bias current. Then a general form for the energy of the network is [10]

$$E = -\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{l=1}^{K} \sum_{i'=1}^{M} \sum_{j'=1}^{M} \sum_{l'=1}^{K} T_{ijl;i'j'l'} V_{ijl} V_{i'j'l'} - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{l=1}^{K} I_{ijl} V_{ijl}. \tag{9}$$

From our discussion in Section II, we note that a solution for the MAP estimate can be obtained by minimizing (8). Here we approximate the posterior energy by

$$U(L \mid Y^*) = \sum_s \left\{ U_1(Y_s^* \mid L_s) + W(L_s) + U_2(L_s) \right\} \tag{10}$$

and the corresponding Gibbs energy to be minimized can be written as

$$E = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{l=1}^{K} U_1(i, j, l)\, V_{ijl} - \frac{\beta}{2} \sum_{l=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{(i',j') \in \hat{N}_{ij}} V_{i'j'l} V_{ijl} \tag{11}$$

where $\hat{N}_{ij}$ is the neighborhood of site (i, j) (same as the $\hat{N}_s$ in Section II). In (11), it is implicitly assumed that each pixel site has a unique label; i.e., only one neuron is active in each column of the network. This constraint can be implemented in different ways. For the deterministic relaxation algorithm described below, a simple method is to use a *winner-takes-all* circuit for each column so that the neuron receiving the maximum input is turned on and the others are turned off. Alternatively, a penalty term can be introduced in (11) to represent the constraint as in [10]. From (9) and (11) we can identify the parameters for the network:

$$T_{ijl;i'j'l'} = \begin{cases} \beta & \text{if } (i', j') \in \hat{N}_{ij}, \forall l = l' \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and the bias current

$$I_{ijl} = -U_1(i, j, l). \tag{13}$$

A. Deterministic Relaxation

The above equations (12) and (13) relate the parameters of the network to that of the image model. The connection matrix for the above network is symmetric and there is no self-feedback; i.e., $T_{ijl;ijl} = 0, \forall i, j, l$. Let u_{ijl} be the potential of neuron (i, j, l). With l the layer number corresponding to texture class l, we have

$$u_{ijl} = \sum_{i'=1}^{M} \sum_{j'=1}^{M} \sum_{l'=1}^{K} T_{ijl;i'j'l'} V_{i'j'l'} + I_{ijl}. \tag{14}$$

In order to minimize (11), we use the following updating rule:

$$V_{ijl} = \begin{cases} 1 & \text{if } u_{ijl} = \min_{l'} \{u_{ijl'}\} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

This updating scheme ensures that at each stage the energy decreases. Since the energy is bounded, the convergence of the above system is ensured but the stable state will in general be a local optimum.

This network model is a version of the iterated conditional mode (ICM) algorithm of Besag [11]. This algorithm maximizes the conditional probability $P(L_s = l \mid Y_s^*, L_{s'}, s' \in \hat{N}_s)$ during each iteration. It is a local deterministic relaxation algorithm that is very easy to implement. We observe that in general an algorithm based on MRF models can be easily mapped onto neural networks with local interconnections. The main advantage of this deterministic relaxation algorithm is its simplicity. Often the solutions are reasonably good and the algorithm usually converges within 20–30 iterations. In the next section we study two stochastic schemes which asymptotically converge to the global optimum of the respective criterion functions.

IV. Stochastic Algorithms for Texture Segmentation

We look at two optimal solutions corresponding to different decision rules for determining the labels. The first one uses simulated annealing to obtain the optimum MAP estimate of the label configuration. The second algorithm minimizes the expected misclassification per pixel. The parallel network implementation of these algorithms is discussed in Section IV-C.

A. Searching for MAP Solution

The MAP rule [18] searches for the configuration L that maximizes the posterior probability distribution. This is equivalent to maximizing $P(Y^* \mid L)\, P(L)$ as $P(Y^*)$ is independent of the labels and Y^* is known. The right-hand side of (8) is a Gibbs distribution. To maximize (8) we use simulated annealing [3], a combinatorial optimization method which is based on sampling from varying Gibbs distribution functions:

$$\frac{\exp\left[-\frac{1}{T_k} U_p(L_s \mid Y_s^*, L_r, r \in \hat{N}_s) \right]}{Z_{T_k}}$$

in order to maximize

$$\frac{e^{-U_p(L \mid Y^*)}}{Z}$$

T_k being the time-varying parameter, referred to as the temperature. We used the following cooling schedule:

$$T_k = \frac{T_0}{1 + \log_2 k} \tag{16}$$

where k is the iteration number. When the temperature is high, the bond between adjacent pixels is loose, and the distribution tends to behave like a uniform distribution over the possible texture labels. As T_k decreases, the distribution concentrates on the lower values of the energy function, which correspond to points with higher probability. The process is bound to converge to a uniform distribution over the label configuration that corresponds to the MAP solution. Since the numer of texture labels is finite, convergence of this algorithm follows from [3]. In our experiment, we realized that starting the iterations with $T_0 = 2$ did not guarantee convergence to the MAP solution. Since starting at a much higher temperature will slow the convergence of the algorithm significantly, we use an alternative approach, viz., cycling the temperature

[13]. We follow the annealing schedule until T_k reaches a lower bound; then we reheat the system and start a new cooling process. By using only a few cycles, we obtained results better than those with a single cooling cycle. Parallel implementation of simulated annealing on the network is discussed in Section IV-C. The results we present in Section VI were obtained with two cycles.

B. *Maximizing the Posterior Marginal Distribution*

The choice of the objective function for optimal segmentation can significantly affect its result. The choice should be made depending on the purpose of the classification. In many implementations the most reasonable objective function is the one that minimizes the expected percentage misclassification per pixel. The solution to the above objective function is also the one that maximizes the marginal posterior distribution of L_s given the observation Y^* for each pixel s:

$$P\{L_s = l_s \mid Y^* = y^*\} \propto \sum_{l \mid L_s = l_s} P(Y^* = y^* \mid L = l)\, P(L = l).$$

The summation above extends over all possible label configurations keeping the label at site s constant. This concept was thoroughly investigated in [14]. Marroquin [19] discusses this formulation in the context of image restoration and illustrates the performance on images with few gray levels. He also mentions the possibility of using this objective function for texture segmentation. In [11] the same objective function is mentioned in the context of image estimation.

To find the optimal solution we use the stochastic algorithm suggested in [14]. The algorithm samples out of the posterior distribution of the texture labels given the intensity. Unlike the stochastic relaxation algorithm, samples are taken with a fixed temperature $T = 1$. The Markov chain associated with the sampling algorithm converges with probability 1 to the posterior distribution. We define new random variables $g_s^{l,t}$ for each pixel ($s \in \Omega$):

$$g_s^{l,t}\{L_s^t\} = \begin{cases} 1 & L_s^t = l \\ 0 & \text{otherwise} \end{cases}$$

where L_s^t is the class of the s pixel, at time t, in the state vector of the Markov chain associated with the Gibbs sampler. We use the ergodic property of the Markov chain [20] to calculate the expectations for these random variables using time averaging:

$$E\{g_s^{l,t}\} = \lim_{N \to \infty} \frac{1}{N} g_s^{l,t} = P_s\{L_s = l \mid Y^*\}$$

where N is the number of iterations performed. To obtain the optimal class for each pixel, we simply chose the class that occurred more often than the others.

The MPM algorithm was implemented using the Gibbs sampler [3]. A much wider set of sampling algorithms, such as Metropolis, can be used for this purpose. The algorithms can be implemented sequentially or in parallel, with a deterministic or stochastic decision rule for the order of visiting the pixels. In order to avoid the dependence on the initial state of the Markov chain, we can ignore the first few iterations. In the experiments conducted we obtained good results after 500 iterations. The algorithm does not suffer from the drawbacks of simulated annealing. For instance we do not have to start the iterations with a high temperature to avoid local minima, and the performance is not badly affected by enlarging the state space.

C. *Network Implementation of the Sampling Algorithms*

All the stochastic algorithms described in the Gibbs formulation are based on sampling from a probability distribution. The probability distribution is constant in the MPM algorithm [14] and is time varying in the case of annealing. The need for parallel implementation is due to the heavy computational load associated with their use.

The issue of parallel implementation in stochastic algorithms was first addressed by Geman and Geman [3]. They show that the Gibbs sampler can be implemented by any deterministic or stochastic rule for choosing the order in which pixels are updated, as long as each pixel is visited infinitely often. An iteration is the time required to visit each pixel at least once (a full sweep). Note that the stochastic rules have a random period and allow us to visit a pixel more than once in a period. They consider the new Markov chain one obtains from the original by viewing it only after each iteration. Their proof is based on two essential elements. The first is the fact that the embedded Markov chain has a strictly positive transition probability p_{ij} for any possible states i, j, which proves that the chain will converge to a unique probability measure regardless of the initial state. The second is that the Gibbs measure is an invariant measure for the Gibbs sampler, so that the embedded chain converges to the Gibbs measure. The proof introduced in [3] can be applied to a much larger family of sampling algorithms satisfying the following properties [20]:

1) The sampler produces a Markov chain with a postive transition probability p_{ij} for any choice of states i, j.
2) The Gibbs measure is invariant under the sampling algorithm.

The Metropolis and heat bath algorithms are two such sampling methods. To see that the Metropolis algorithm satisfies property 2, we look at the following equation for updating a single pixel:

$$P^{n+1}(i) = \frac{1}{m} \sum_{\pi(j) < \pi(i)} P^n(j) + \frac{1}{m} \sum_{\pi(j) < \pi(i)} P^n(i)\, \frac{\pi(i) - \pi(j)}{\pi(i)} + \frac{1}{m} \sum_{\pi(j) \geq \pi(i)} P^n(j)\, \frac{\pi(i)}{\pi(j)}$$

where m is the number of values each pixel can take. The first term corresponds to the cases when the system was in state j and the new state i has higher probability. The second term corresponds to a system in state i and a new state j that has lower probability. The given probability is for staying in state i. The third term corresponds to a system in state j and a new state i with lower probability. If we now replace $P^{n+1}(i)$ and $P^n(i)$ with $\pi(i)$ and $P^n(j)$, we see that the equality holds, implying that the Gibbs measure is invariant under the Metropolis algorithm. The first property is also satisfied. Note that the states now correspond to the global configuration. To implement the algorithm in parallel, one can update pixels in parallel as long as neighboring pixels are not updated at the same time. A very clear discussion on this issue can be found in [14].

We now describe how these stochastic algorithms can be implemented on the network discussed in Section III. The only modification required for the simulated annealing rule is that the neurons in the network fire according to a time-dependent probabilistic rule. Using the same notation as in Section III, the probability that neuron (i, j, l) will fire during iteration k is

$$P(V_{ijl} = 1) = \frac{e^{-(1/T_k)u_{ijl}}}{Z_{T_k}} \tag{17}$$

where u_{ijl} is as defined in (14) and T_k follows the cooling schedule (16).

The MPM algorithm uses the above selection rule with $T_k = 1$. In addition, each neuron in the network has a counter which is incremented every time the neuron fires. When the iterations are terminated the neuron in each column of the network having the maximum count is selected to represent the label for the corresponding pixel site in the image.

We have noted before that for parallel implementation of the sampling algorithms, neighboring sites should not be updated simultaneously. Some additional observations are made in Section VI.

V. Stochastic Learning and Neural Networks

In the previous sections purely deterministic and stochastic relaxation algorithms were discussed. Each has its own advantages and disadvantages. Here we consider the possibility of combining the two methods using stochastic learning automata and we compare the results obtained by this new scheme with those of previous algorithms.

We begin with a brief introduction to the stochastic learning automaton [12]. An automation is a decision maker operating in a random environment. A stochastic automation can be defined by a quadruple (α, Q, T, R), where $\alpha = \{\alpha_1, \cdots, \alpha_N\}$ is the set of available actions to the automaton. The action selected at time t is denoted by $\alpha(t)$. $Q(t)$ is the state of the automaton at time t and consists of the action probability vector $p(t) = [p_1(t), \cdots, p_N(t)]$, where $p_i(t) = \text{prob}(\alpha(t) = \alpha_i)$ and $\Sigma_i p_i(t) = 1 \ \forall t$. The environment responds to the action $\alpha(t)$ with a $\lambda(t) \in R$, R being the set of the environment's responses. The state transitions of the automaton are governed by the learning algorithm T, $Q(t+1) = T(Q(t), \alpha(t), \lambda(t))$. Without loss of generality, it can be assumed that $R = [0, 1]$; i.e., the responses are normalized to lie in the interval $[0, 1]$, 1 indicating a complete *success* and 0 total *failure*. The goal of the automaton is to converge to the optimal action, i.e., the action which results in the maximum expected reward. Again without loss of generality let α_1 be the optimal action and $d_1 = E[\lambda(t) \mid \alpha_1] = \max_i \{E[\lambda(t) \mid \alpha_i]\}$. At present no learning algorithms exist which are optimal in the above sense. However we can choose the parameters of certain learning algorithms so as to realize a response as close to the optimum as desired. This condition is called ϵ optimality. If $M(t) \triangleq E[\lambda(t) \mid p(t)]$, then a learning algorithm is said to be ϵ optimal if it results in an $M(t)$ such that

$$\lim_{t \to \infty} E[M(t)] > d_1 - \epsilon \tag{18}$$

for a suitable choice of parameters and for any $\epsilon > 0$. One of the simplest learning schemes is the linear reward-inaction rule, L_{R-I}. Suppose at time t we have $\alpha(t) = \alpha_i$; if $\lambda(t)$ is the response received, then according to the L_{R-I} rule

$$p_i(t+1) = p_i(t) + a\lambda(t)[1 - p_i(t)]$$
$$p_j(t+1) = p_j(t)[1 - a\lambda(t)], \quad \forall j \neq i \tag{19}$$

where a is a parameter of the algorithm controlling the learning rate. Typical values for a are in the range 0.01–0.1. It can be shown that this L_{R-I} rule is ϵ optimal in all stationary environments; i.e., there exists a value for the parameter a so that condition (18) is satisfied.

Collective behavior of a group of automata has also been studied. Consider a team of N automata $A_i (i = 1, \cdots, N)$, each having r_i actions $\alpha^i = \{\alpha_1^i \cdots \alpha_{r_i}^i\}$. At any instant t each member of the team makes a decision $\alpha^i(t)$. The environment responds to this by sending a reinforcement signal $\lambda(t)$ to all the automata in the group. This situation represents a cooperative game among a team of automata with an identical payoff. All the automata update their action probability vectors according to (19) using the same learning rate, and the process repeats. Local convergence results can be obtained for the case of stationary random environments. Variations of this rule have been applied to complex problems such as decentralized control of Markov chains [21] and relaxation labeling [22].

The texture classification discussed in the previous sections can be treated as a relaxation labeling problem and stochastic automata can be used to learn the labels (texture class) for the pixels. A learning automaton is assigned to each of the pixel sites in the image. The actions of the automata correspond to selecting a label for the pixel site to which it is assigned. Thus for a K-class problem each automaton has K actions and a probability distribution over this action set. Initially the labels are assigned randomly with equal probability. Since the number of automata involved is very large, it is not practical to update the action probability vector at each iteration. Instead we combine the iterations of the neural network described in the previous section with the stochastic learning algorithm. This results in an iterative hill-climbing-type algorithm which combines the fast convergence of deterministic relaxation with the sustained exploration of the stochastic algorithm. The stochastic part prevents the algorithm from getting trapped in local minima and at the same time "learns" from the search by updating the state probabilities. However, in contrast to simulated annealing, we cannot guarantee convergence to the global optimum. Each cycle now has two phases: the first consists of the deterministic relaxation network converging to a solution; the second consists of the learning network updating its state, the new state being determined by the equilibrium state of the relaxation network. A new initial state is generated by the learning network depending on its current state and the cycle repeats. Thus relaxation and learning alternate with each other. After each iteration the probability of the more stable states increases and because of the stochastic nature of the algorithm the possibility of getting trapped in bad local minima is reduced. The algorithm is summarized below.

A. Learning Algorithm

Let the pixel site be denoted by $s \in \Omega$ and the number of texture classes be K. Let A_s be the automaton assigned to site s and the action probability vector of A_s be $p_s(t) = [p_{s,1}(t), \cdots, p_{s,k}(t)]$ and $\Sigma_i p_{s,i}(t) = 1 \; \forall s, t$, where $p_{s,l}(t) = \text{prob}$ (label of site $s = l$). The steps in the algorithm are as follows:

1) Initialize the action probability vectors of all the automata

$$p_{s,l}(0) = 1/K, \qquad \forall s, l.$$

Initialize the iteration counter to 0.

2) Choose an initial label configuration sampled from the distribution of these probability vectors.
3) Start the neural network of Section III with this configuration.
4) Let l_s denote the label for site s at equilibrium. Let the current time (iteration number) be t. Then the action probabilities are updated as follows:

$$p_{s,l_s}(t+1) = p_{s,l_s}(t) + a\lambda(t)[1 - p_{s,l_s}(t)]$$
$$p_{s,j}(t+1) = p_{s,j}(t)[1 - a\lambda(t)], \qquad \forall j \neq l_s \text{ and } \forall s. \tag{20}$$

The response $\lambda(t)$ is derived as follow: Suppose the present label configuration resulted in a lower energy state than the previous one. Then it results in $\lambda(t) = \lambda_1$, and if the energy increases we have $\lambda(t) = \lambda_2$ with $\lambda_1 > \lambda_2$. In our simulations we used $\lambda_1 = 1$ and $\lambda_2 = 0.25$.

5) Generate a new configuration from this updated label probabilities, increment the iteration counter, and go to step 3.

Thus the system consists of two layers, one for relaxation and the other for learning. The relaxation network is similar to the one considered in Section III, the only difference being that the initial state is decided by the learning network. The learning network consists of a team of automata and learning takes place at a much lower speed than the relaxation, with fewer updatings. The probabilities of the labels corresponding to the final state of the relaxation network are increased according to (20). Using these new probabilities a new configuation is generated. Since the response does not depend on time, this corresponds to a stationary environment, and as we have noted before this L_{R-I} algorithm can be shown to converge to a stationary point, not necessarily to the global optimum.

VI. Experimental Results and Conclusions

The segmentation results using the above algorithms are given on two examples. The parameters σ_l and Θ_l corresponding to the fourth-order GMRF for each texture class were precomputed from 64×64 images of the textures. The local mean (in an 11×11 window) was first subtracted to obtain the zero mean texture, and the least-square estimates [17] of the parameters were then computed from the interior of the image. The first step in the segmentation process involves computing the Gibbs energies $U_1(Y_s^* \mid L_s)$ in (3). This is done for each texture class and the results are stored. For computational convenience these $U_1(\cdot)$ values are normalized by dividing by k^2, where k is the size of the window. To ignore the boundary effects, we set $U_1 = 0$ at the boundaries. We have experimented with different window sizes; larger windows result in more homogeneous texture patches but the boundaries between the textures are distorted. The results reported here are based on windows of size 11×11 pixels. The bias term $w(l_s)$ can be estimated using the histogram of the image data [18] but we obtained these values by trial and error.

In Section IV we observed that neighboring pixel sites should not be updated simultaneously. This problem occurs only if digital implementations of the networks are considered, as the probability of this happening in an analog network is zero. When this simultaneous updating was tested for the deterministic case, it always converged to limit cycles of length 2. (In fact it can be shown that the system converges to limit cycles of length at most 2.)

The choice of β plays an important role in the segmentation process and its value depends on the magnitude of the energy function $U_1(\cdot)$. Various values of β ranging from 0.2–3.0 were used in the experiments. In the deterministic algorithm it is preferable to start with a small β and increase it gradually. Large values of beta usually degrade the performance. We also observed that slowly increasing β during the iterations improves the results for the stochastic algorithms. It should be noted that using a larger value of β for the deterministic algorithm (compared to those used in the stochastic algorithms) does not improve the performance.

The nature of the segmentation results depends on the order of the label model. It is preferable to choose the first-order model for the stochastic algorithms if we know *a priori* that the boundaries are either horizontal or vertical. However, for the deterministic rule and the learning scheme the second-order model results in more homogeneous classification.

The MPM algorithm requires the statistics obtained from the invariant measure of the Markov chain corresponding to the sampling algorithm. Hence it is preferable to ignore the first few hundred trials before starting to gather the statistics. The performance of the deterministic relaxation rule of Section III also depends on the initial state and we have looked into two different initial conditions. The first one starts with a label configuration L such that $L_s = l_s$ if $U_1(Y_s^* \mid l_s) = \min_{l_k} \{U_1(Y_s^* \mid l_k)\}$. This corresponds to maximizing the probability $P(Y^* \mid L)$ [23]. The second choice for the initial configuration is a randomly generated label set. Results for both cases are provided and we observe that the random choice often leads to better results.

In the examples below the following learning parameters were used: learning rate $a = 0.05$ and reward/penalty parameters $\lambda_1 = 1.0$ and $\lambda_2 = 0.25$.

Example 1: This is a two-class problem consisting of grass and calf textures. The image is of size 128×128 and is shown in Fig. 2(a). In Fig. 2(b) the classification obtained by the deterministic algorithm discussed in Section III is shown. The maximum likelihood estimate was the initial state for the network, and Fig. 2(c) gives the result with random initial configuration. Notice that in this case the final result has fewer misclassified regions than in Fig. 2(b) and this was observed to be true in general. Parts (d) and (e) of the figure give the MAP solution using simulated annealing and the MPM solution respectively. The result of the learning algorithm is shown in Fig. 2(f) and there are no misclassifications within the homogeneous regions. However the boundary is not as good as those of the MAP or MPM solutions. In all the cases we used $\beta = 0.6$.

Example 2: This is a 256×256 image (Fig. 2(a)) having six textures: calf, grass, wool, wood, pig skin, and sand. This is a difficult problem in the sense that three of the textures (wool, pig skin, and sand) have almost identical characteristics and are not easily distinguishable, even by the human eye. The maximum likelihood solution is shown in Fig. 3(b), and part (c) of the figure is the solution obtained by the deterministic relaxation network with the result in part (b) as the initial condition. Fig. 3(d) gives the result with random initial configuration. The MAP solution using simulated annealing is shown in part (e). As was mentioned in Section IV-A, cycling of temperature improves the performance of simulated annealing. The segmentation result ws obtained by starting with an initial temperature $T_0 = 2.0$ and cooling according to the schedule (16) for 300 iterations. Then the system was reset to $T_0 = 1.5$ and the process was repeated for 300 more iterations. In the case of the MPM rule the first 500 iterations were ignored and Fig. 3(f) shows the result obtained using the last 200 iterations. As in the previous example, the best results were obained by the simulated annealing and MPM algorithms. For the MPM case there were no misclassifications within homogeneous regions but the boundaries were not accurate. In fact, as indicated in Table I, simulated annealing has the lowest percentage error in classification. Introducing learning in deterministic relaxation considerably improves the performance (Fig. 3(g)). Table I gives the percentage classification error for the different cases.

It is noted from the table that although learning improves the performance of the deterministic network algorithm, the best results were obtained by the simulated annealing technique, which is to be expected.

A. Hierarchical Segmentation

The various segmentation algorithms described in the previous sections can be easily extended to hierarchical structures wherein the segmentation is carried out at different levels—from coarse to fine. The energy functions are modified to take care of the coupling between the adjacent resolutions of the system. Consider a K-stage hierarchical system, with stage 0 representing the maximum resolution level and stage $K - 1$ being the coarsest level. The energy corresponding to the kth stage is denoted by $U_1^k(s, l)$ and $U_2^k(s)$ (eqs. (3) and (4)). The size of the window used in computing the joint energy potential $U_1^k(\cdot)$ increases with the index k. The potential U_2 is modified to take care of the coupling as follows:

$$U_2^k(s) = -\beta \sum_{t \in D_s^k} \delta(L^k(s) - L^k(t)) + \beta_k\big(\delta(L^k(s) - L^{k+1}(s))\big) + w(L(s)), \quad 0 \le k < K - 1 \tag{21}$$

where $L^k(s)$ is the label for the site s in the kth stage, and β_k is the coupling coefficient between the stages $k + 1$ and k. $D^k(s)$ is the appropriate neighborhood set for the kth stage. The result of segmentation on the six-class problem with $K = 2$ and using the learning algorithm is shown in Fig. 3(h).

B. Conclusion

In this paper we have looked into different texture segmentation algorithms based on modeling the texture intensities as a GMRF. It is observed that a large class of natural textures can be modeled in this way. The perfor-

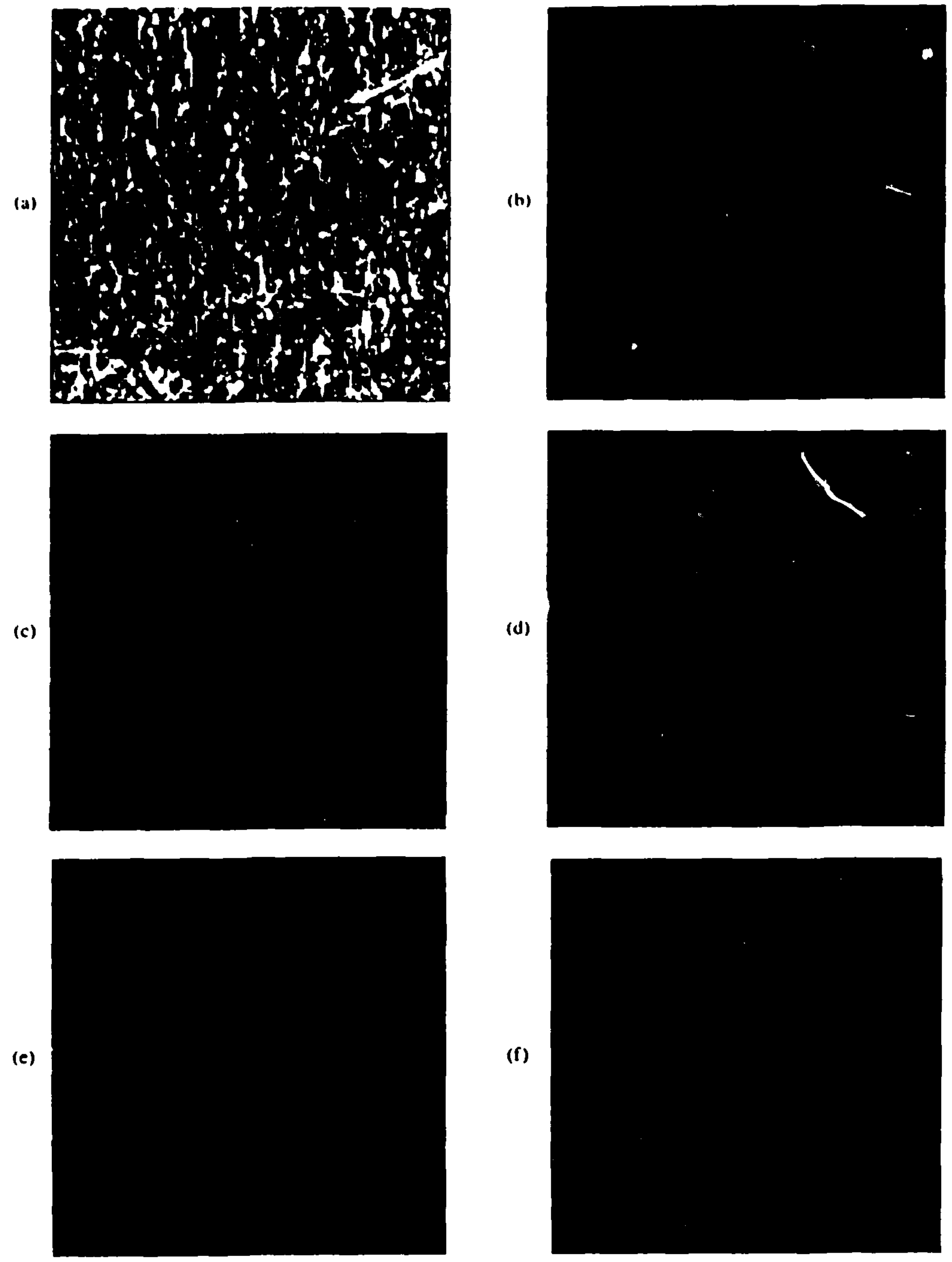


Fig. 2. (a) Original image consisting of two textures. The classification using different algorithms is shown in (b)-(f). (The textures are coded by gray levels.) (b) Deterministic relaxation with maximum likelihood solution as initial condition and (c) with random initial condition. (d) MAP estimate using simulated annealing. (e) MPM solution. (f) Network with stochastic learning.

mance of several algorithms for texture segmentation is studied. The stochastic algorithms obtain nearly optimal results, as can be seen from the examples. We noted that the MRF model helps us to trivially map the optimization problem onto a Hopfield-type neural network. This deterministic relaxation network converges extremely fast to a solution, typically in 20-30 iterations for the 256 × 256 image. Its performance, however, is sensitive to the initial state of the system and often is not very satisfactory. To overcome the disadvantages of the network, a new algorithm, which introduces stochastic learning into the iterations of the network, was proposed. This helps to maintain a sustained search of the solution space while learning from the past experience. This algorithm combines the advantages of deterministic and stochastic relaxation schemes and it would be interesting to explore its performance in solving other computationally hard problems in computer vision.

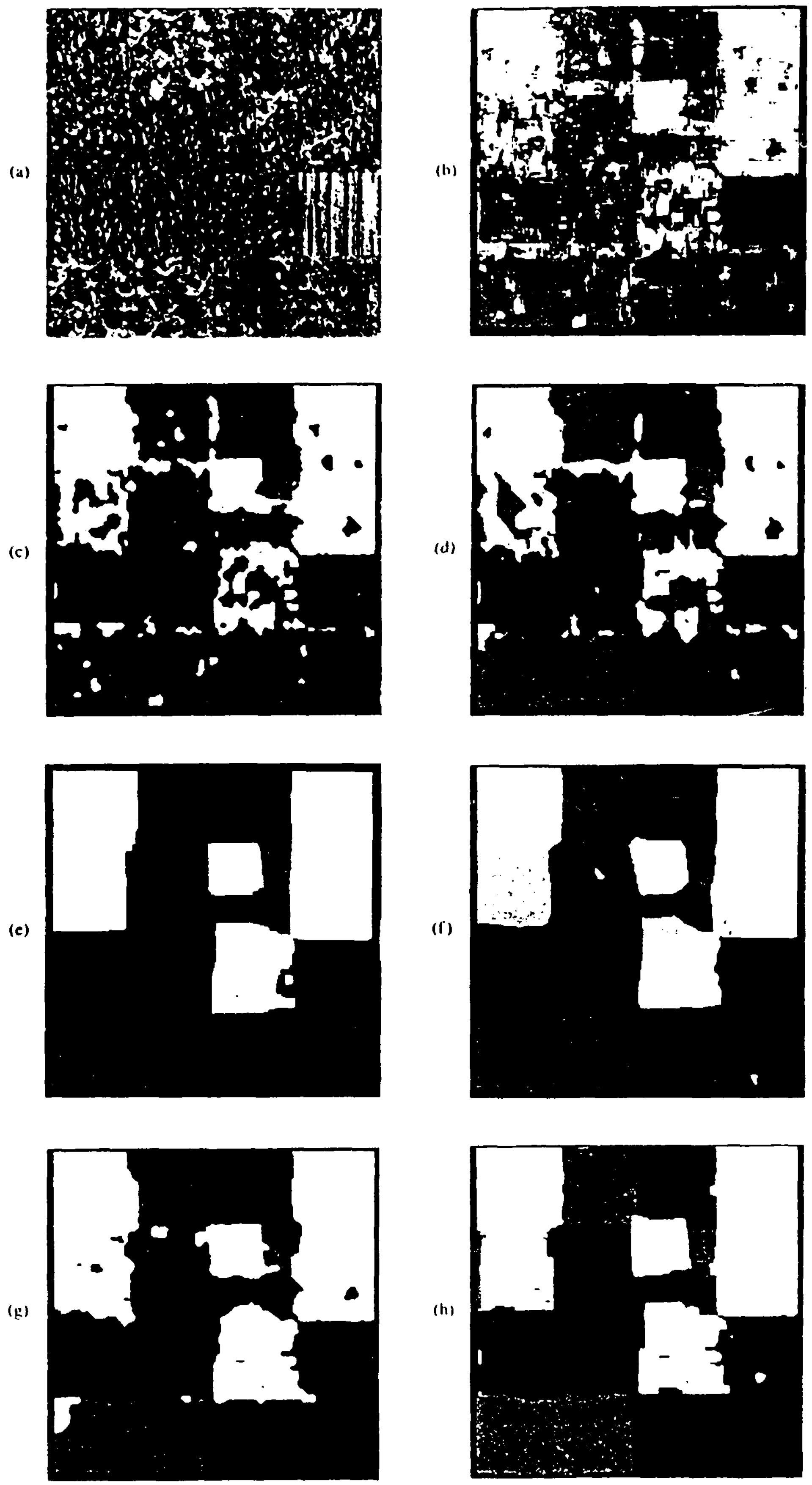


Fig. 3. (a) Original image consisting of six textures. (b) Maximum likelihood solution. (c) Deterministic relaxation with (b) as initial condition and (d) with random initial condition. (e) MAP estimate using simulated annealing. (f) MPM solution. (g) Network with stochastic learning. (h) Hierarchical network solution.

TABLE I
PERCENTAGE MISCLASSIFICATION FOR EXAMPLE 2 (SIX CLASS PROBLEM)

Algorithm	Percentage Error
Maximum Likelihood Estimate	22.17
Neural network (MLE as initial state)	16.25
Neural network (Random initial state)	14.74
Simulated annealing (MAP)	6.72
MPM algorithm	7.05
Neural network with learning	8.7
Hierarchical network	8.21

ACKNOWLEDGMENT

The authors wish to thank the reviewers for the many useful comments and suggestions which helped to improve the presentation of this paper.

REFERENCES

[1] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 39-55, Jan. 1987.
[2] F. S. Cohen and D. B. Cooper, "Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian fields," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 195-219, Mar. 1987.
[3] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721-741, Nov. 1984.
[4] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314-319, Sept. 1985.
[5] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, pp. 1141-1151, July 1988.
[6] Y. T. Zhou and R. Chellappa, "Stereo matching using a neural network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* (New York, NY), Apr. 1988, pp. 940-943.
[7] C. Koch, J. Luo, C. Mead, and J. Hutchinson, "Computation motion using resistive networks," in *Proc. Neural Inform. Process. Syst.* (Denver, CO), 1987.
[8] Y. T. Zhou and R. Chellappa, "Computation of optical flow using a neural network," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2 (San Diego, CA), pp. 71-78.
[9] H. Bulthoff, J. Little, and T. Poggio, "A parallel algorithm for real-time computation of optical flow," *Nature*, vol. 337, pp. 549-553, Feb. 1989.
[10] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biolog. Cybernet.*, vol. 52, pp. 114-152, 1985.
[11] J. Besag, "On the statistical analysis of dirty pictures," *J. Roy. Statist. Soc. B*, vol. 48, pp. 259-302, 1986.
[12] K. S. Narendra and M. A. L. Thathachar, "Learning automata—A survey," *IEEE Trans. Syst., Man, Cybern.*, pp. 323-334, July 1974.
[13] U. Grenander, *Lectures in Pattern Theory*, vols. I-III. New York: Springer-Verlag, 1981.
[14] J. L. Marroquin, "Probabilistic solution of inverse problems," Ph.D. thesis, M.I.T., Artificial Intelligence Laboratory, Sept. 1985.
[15] R. Chellappa, "Two dimensional discrete Gaussian Markov random field models for image processing," in *Progress in Pattern Recognition 2*, L. N. Kanal and A. Rosenfeld, Eds. New York: Elsevier, 1985, pp. 79-112.
[16] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 25-39, Jan. 1983.
[17] R. Chellappa and S. Chatterjee, "Classification of textures using Gaussian Markov random fields," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-33, pp. 959-963, Aug. 1985.
[18] S. Geman and C. Graffigne, "Markov random fields image models and their application to computer vision," in *Proc. Int. Congress of Mathematicians 1986* (Providence).
[19] J. Marroquin, S. Mitter, and T. Poggio, "Probabilistic solution of ill-posed problems in computer vision," in *Proc. Image Understanding Workshop* (Miami Beach, FL), Dec. 1985, pp. 293-309.
[20] B. Gidas, "Non-stationary Markov chains and convergence of the annealing algorithm," *J. Statist. Phys.*, vol. 39, pp. 73-131, 1985.
[21] R. M. Wheeler, Jr., and K. S. Narendra, "Decentralized learning in finite Markov chains," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 519-526, June 1986.
[22] M. A. L. Thathachar and P. S. Sastry, "Relaxation labeling with learning automata," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 526-268, Mar. 1986.
[23] S. Chatterjee and R. Chellappa, "Maximum likelihood texture segmentation using Gaussian Markov random field models," in *Proc. Computer Vision and Pattern Recognition Conf.* (San Francisco, CA), June 1985.

B. S. Manjunath (S'88) received the bachelor of engineering degree in electronics from Bangalore University in 1985, and the master of engineering degree in systems science and automation from the Indian Institute of Science in 1987. Since 1987 he has been a Research Assistant at the Signal and Image Processing Institute, University of Southern California, Los Angeles, where he is currently working toward the Ph.D. degree in electrical engineering. His research interests include stochastic learning, self-organization, neural networks, and computer vision.

Tal Simchony (S'86–M'89) was born in Tel Aviv, Israel, on January 18, 1956. He received the B.S. degree in mathematics and computer science and the M.S. degree in applied mathematics from Tel Aviv University in 1982 and 1985, respectively. He then received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1988.

In 1982 he joined ECI Telecom as a Software and Systems Engineer. During the years 1985–1988 he was a Research Assistant at the Signal and Image Processing Institute, USC. He is currently at ECI Telecom as Deputy Chief Engineer working on speech compression algorithms on digital networks. His research interests include optimization, learning, and computer vision.

Rama Chellappa (S'79–M'81–SM'83) was born in Madras, India. He received the B.S. degree (honors) in electronics and communications engineering from the University of Madras in 1975 and the M.S. degree (with distinction) in electrical communication engineering from the Indian Institute of Science in 1977. He then received the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1978 and 1981, respectively.

During the years 1979–1981, he was a Faculty Research Assistant at the Computer Vision Laboratory, University of Maryland, College Park. Since 1986, he has been an Associate Professor in the Electrical Engineering-Systems, University of Southern California, Los Angeles, and in September 1988 he became the Director of the Signal and Image Institute there. His current research interests are in signal and image processing, computer vision, and pattern recognition.

Dr. Chellappa is a member of Tau Beta Pi and Eta Kappa Nu. He is a coeditor of two volumes of selected papers on image analysis and processing, published in the autumn of 1985. He was an Associate Editor for the IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING and he is a coeditor of *Computer Vision, Graphics, and Image Processing: Graphic Models and Image Processing*, published by Academic Press. He was a recipient of a National Scholarship from the Government of India during the period 1969–1975. He was the recipient of the 1975 Jawaharlal Nehru Memorial Award from the Department of Education, Government of India, the 1985 Presidential Young Investigator Award, and the 1985 IBM Faculty Development Award. He served as the General Chairman of the 1989 IEEE Computer Society Conference on Computer Vision and Pattern Recognition and the IEEE Computer Society Workshop on Artificial Intelligence for Computer Vision. He was also Program Cochairman of the NSF sponsored Workshop on Markov Random Fields for Image Processing Analysis and Computer Vision.

IEEE Transactions on Patt., Anal., Mach. Intell., (In revision).

A Note on Unsupervised Texture Segmentation [1]

B.S. Manjunath and R. Chellappa

Department of EE-Systems

University of Southern California

Los Angeles, California 90089

Abstract

We consider the problem of unsupervised segmentation of textured images. The only explicit assumption made is that the intensity data can be modeled by a Gauss Markov Random Field (GMRF). The image is divided into number of non-overlapping regions and the GMRF parameters are computed from each of these regions. A simple clustering method is used to merge these regions. The parameters of the model estimated from the clustered segments are then used in two different schemes, one being an approximation to the maximum aposteriori estimate of the labels and the other minimizing the percentage misclassification error. Our approach is contrasted with a recently published algorithm [1] which detailed an interesting simultaneous parameter estimation and segmentation scheme. We compare the results of the adaptive segmentation algorithm in [1] with a simple nearest neighbor classification scheme to show that if enough information is available, simple techniques could be used as alternatives to computationally expensive schemes.

[1]Partially supported by the AFSOR grant 86-0196.

1 Introduction

Segmenting a textured scene into different classes in the absence of apriori information is still an unsolved issue in computer vision. The main difficulty is that the model and its parameters are unknown and need to be computed from the given image before segmentation. To compute the parameters effectively the segmented image itself is needed ! Simultaneous parameter estimation and segmentation is often computationally prohibitive. An alternate approach to this problem is to have a two step process, first estimating the parameters in small regions and getting a crude segmentation. Then estimate the parameters again from this segmented image and use pixel based segmentation schemes ([2],[3]). In this paper we assume that the texture intensity distribution can be modeled by a second order GMRF. Hence the problem is in estimating these GMRF parameters and segmenting the textures based on the estimated values.

Unsupervised texture segmentation is not a new problem. Some of the recent work has been reported in [1], [4] and [5]. Lakshmanan and Derin [1] in a recent paper address the problem of simultaneous estimation and segmentation of Gibbs Random Fields (GRF). They obtained an interesting convergence result for the Maximum Likelihood Estimates (MLE) of the parameters and maximum aposteriori probability (MAP) solution for the segmentation. We give a brief description of their model in section 5 and experimental results to illustrate that if one makes the same assumptions, a simple nearest neighbor classification rule produces results very close to those obtained using simulated annealing as in [1]. In [4], no specific texture model is assumed. Certain features are extracted from the sub-images and the image is segmented based on the disparity measure between the feature vectors from different sub-images.

The approach to texture segmentation presented here is similar to the work of Cohen and Fan [5]. In [5] the textures are modeled as second order GMRF and the texture parameters are estimated from disjoint windows. The windows are later grouped based on clustering analysis. Finer segmentation is obtained by using the parameters from the coarse segmentation in a suitable relaxation algorithm [6].

In the next section we give a brief description of the texture model. Section 3 details the segmentation scheme and the experimental results are provided in section 4. In section 5, the adaptive segmentation scheme of [1] is discussed along with the results of a simple nearest neighbor classification scheme.

2 Texture model

The GMRF model for textures has been used by many researchers [7]. In this paper we consider a second order GMRF model for the conditional probability density of the intensity given the texture label.

Let Ω denote the set of grid points in the $M \times M$ lattice, i.e., $\Omega = \{(i,j)\ ,\ 1 \leq i,j \leq M\}$. Let $\{L_s\ ,\ s \in \Omega\}$ and $\{Y_s\ ,\ s \in \Omega\ \}$ denote the labels and zero mean gray level arrays respectively. Let N_s be the symmetric second order neighborhood of a site s (consisting of the 8 nearest neighbors of s). Then assuming that all the neighbors of s also have the same label as that of s, we can write the following expression for the conditional density of the intensity at the pixel site s:

$$P(Y_s = y_s \mid Y_r = y_r, r \in N_s, L_s = l) = \frac{e^{-U(Y_s=y_s \mid Y_r=y_r, r\in N_s, L_s=l)}}{Z(l|y_r, r \in N_s)} \tag{1}$$

where $Z(l|y_r, r \in N_s)$ is the partition function of the conditional Gibbs distribution and

$$U(Y_s = y_s \mid Y_r = y_r; r \in N_s, L_s = l) = \frac{1}{2\sigma_l^2}(y_s^2 - 2\sum_{r\in N_s} \Theta^l_{s,r} y_s y_r) \tag{2}$$

In (2), σ_l and Θ^l are the GMRF model parameters of the l-th texture class. The model parameters satisfy $\Theta^l_{r,s} = \Theta^l_{r-s} = \Theta^l_{s-r} = \Theta^l_r$.

Further, the joint probability in a window W_s centered at s can be written as,

$$P(\mathbf{Y}^*_s | L_s = l) = \frac{e^{-U_1(\mathbf{y}^*_s | L_s = l)}}{Z_1(l)}$$

where $Z_1(l)$ is the partition function and

$$U_1(\mathbf{y}_s^*|L_s = l) = \frac{1}{2\,\sigma_l^2} \sum_{r \in W_s} \left\{ y_r^2 - \sum_{\tau \in N^* | r+\tau \in W_s} \Theta_\tau^l y_r (y_{r+\tau} + y_{r-\tau}) \right\} \tag{3}$$

$\mathbf{y}_s^*$ represents the intensity array in the window W_s. The above equation assumes a free boundary model. N^* is a set of shift vectors corresponding to the second order GMRF model,

$$\begin{aligned} N^* &= \{\tau_1, \tau_2, \tau_3, \tau_4\} \\ &= \{(0,1),(1,0),(1,1),(-1,1)\} \end{aligned} \tag{4}$$

2.1 GMRF Parameter Estimation

There are many existing methods for estimating the GMRF parameters, but none of them can guarantee both consistency (estimates converging to the true values of the parameters) and stability (the covariance matrix in the expression for the joint probability density of the MRF must be positive definite) together. Normally an optimization algorithm is run to obtain the stable estimates. Here we consider the least square estimates of the GMRF parameters [8]. Since our main interest is in obtaining reasonably good measures to aid the segmentation process and not in synthesizing the textures, we do not check for the stability of the estimates obtained. One can instead use the maximum likelihood estimates [9], but it is computationally more expensive. Consider a region of size $N \times N$ containing a single texture. Let Ω be the lattice under consideration and let Ω_I be the interior region of Ω, i.e.,

$$\Omega_I = \Omega - \Omega_B, \Omega_B = \{s = (i,j), s \in \Omega \text{ and } s \pm \tau \notin \Omega \text{ for at least some } \tau \in N^*\} \tag{5}$$

Let

$$Q_s = [y_{s+\tau_1} + y_{s-\tau_1}, \ldots, y_{s+\tau_4} + y_{s-\tau_4}]^T \tag{6}$$

Then the least square estimates of the parameters are

$$\hat{\Theta} = [\sum_{\Omega_I} Q_s Q_s^T]^{-1} [\sum_{\Omega_I} Q_s y_s] \tag{7}$$

$$\hat{\sigma}^2 = \frac{1}{N^2} \sum_{\Omega_I} [y_s - \Theta^T Q_s]^2 \tag{8}$$

If μ is the mean of the subimage, then the feature vector for the region is denoted by

$$\mathbf{F} = (f_1, f_2, f_3, f_4, f_5, f_6) = (\theta_1, \theta_2, \theta_3, \theta_4, \mu, \sigma^2) \tag{9}$$

Label field : The label field is modeled as a second order discrete MRF. It does not play any role in parameter estimation or in obtaining the initial coarse segmentation. The label field is characterized by a single parameter β which determines the bonding between different regions in the image.

3 Segmentation

3.1 Clustering

The given image is divided in to a number of non-overlapping subimages. For each of these subimages the corresponding feature vectors are estimated as described in the previous section. It is assumed that all these subimages are homogeneous. A normalized Eucledian distance measure is defined for these vectors as

$$d(F^i, F^j) = \sum_k \frac{(f_k^i - f_k^j)^2}{(f_k^i)^2 + (f_k^j)^2} \tag{10}$$

A simple clustering is done based on this distance measure. First the maximum distance between any two regions in the image is found as

$$d_{max} = \max_{i,j} d(F^i, F^j)$$

The regions are now grouped such that any two subimages i and j belonging to the same class satisfy

$$d(F^i, F^j) < \rho \;\; d_{max} \tag{11}$$

where ρ is a clustering parameter. Since ρ affects the number of clusters that are formed, a good guess of ρ should be based on the knowledge about the approximate number of classes present in the image. In our experiments we used a simple heuristic $\rho = 1/$(approximate number of classes). In the above clustering process all isolated regions are marked as ambiguous. Also all regions which satisfy the criterion (11) for two different classes should be labelled ambiguous. Usually the boundary regions which have more than one texture inside fall into this class. Note that alternate schemes like k-mean clustering can also be used in obtaining such a coarse segmentation.

From these clustered regions the parameters are recomputed. These parameters are then used in pixel based segmentation algorithms [3] to obtain finer segmentation.

3.2 Deterministic and stochastic algorithms

3.2.1 Deterministic relaxation

Assuming that the parameters of the model and the number of classes in the image are known, the texture segmentation problem can be formulated as a minimization problem. Further, for the case when the model is a MRF, mapping this problem on to a relaxation network is straightforward. The function to be minimized can be written as [3]

$$E = \frac{1}{2}\sum_{s}\sum_{l=1} U(s,l)V_{sl} - \frac{\beta}{2}\sum_{l=1}\sum_{s}\sum_{s'\in N_s} V_{s'l}V_{sl} \tag{12}$$

where N_s is the second order neighborhood of site s and $\{V_{sl}\}$ are variables taking on values from $\{0,1\}$. If V_{sl} is 1, it indicates that the site s belongs to class l. Note that for each s, only one V_{sl} has a value one and all others are zero. β represents the binding between textures of the same class and characterizes the initial distribution of the class labels. $U(s,l)$ includes all the information regarding the intensity and parameter values for the site $s \in$ class l. It gives a measure of the joint distribution of the intensities in a small window W_s centered at s and for the case when all pixels inside the window belong to class l, it is given by

$$U(s,l) = w(l) + U_1(Y_s^*, l) \tag{13}$$

where $U_1(.)$ is as in (3) and $w(l)$ is the bias corresponding to class l [3]. This bias can be estimated from the given data as we have a coarse initial segmentation to begin with. N^* is the set of shift vectors corresponding to second order GMRF model as in (4). Before starting the relaxation, we can selectively fix the labels of the pixels from which the parameters are initially estimated, so that the relaxation process can be faster.

During each visit to site s, the class corresponding to the lowest energy E is selected. This is equivalent to setting the appropriate V_{sl} to 1. The process is repeated till there is no change in the energy E, i.e. till convergence. It can be shown that this relaxation converges to a solution, which may not be the best always. This algorithm is similar to the iterated conditional mode rule proposed by Besag [10].

3.2.2 Stochastic algorithms

The alternative to deterministic relaxation is to update the class labels in a random way. Simulated annealing can be used to get the MAP solution [2]. Here we consider another criterion which minimizes the expected classification error per pixel (or alternatively, maximizes the posterior marginal distribution) and use the algorithm suggested in [11] for this. This algorithm is equivalent to running simulated annealing at a fixed temperature T = 1 (i.e., no annealing) and for details we refer to [3]. The final labels chosen correspond to the most frequently selected ones. For convenience we refer to this as the MPM (Maximizing the Posterior Marginal) algorithm in the following. We also implemented an algorithm which combines the deterministic relaxation with stochastic learning [3]. This has an advantage that it requires fewer number of iterations compared to simulated annealing and the results are better than using the deterministic relaxation alone. Learning is introduced by defining a probability distribution over the class labels at each pixel site and these probabilities are updated at each convergence of the deterministic relaxation. A new starting state for the relaxation is obtained by sampling from this updated probability distribution and the process is repeated. Usually about 20-40 such learning cycles are enough to get good results.

4 Experimental Results

In the experiments described below, the subimage size was chosen to be 32x32. The value of the clustering parameter depends on the number of texture classes present and as mentioned earlier we used the heuristic $\rho = 1/$(approximate number of classes). To eliminate very small isolated regions one can use a penalty function in the relaxation algorithm which prohibits small clusters from being formed [4]. We found it convenient to use a smoothing filter (size 3x3 or 5x5) to do the same. An useful observation is that with this kind of "post-processing", the performance of both the deterministic and stochastic relaxation algorithms is comparable. The results given below correspond to those obtained after performing the smoothing. However the boundaries obtained by the stochastic algorithms are more accurate.

Example 1: (Grass and leather texture) Figure 1 shows this mosaic and is a 128 x 128 image and $\rho = 0.5$. Figure 1(b) shows the result of coarse clustering described in section 3.1. Figure 1(c) is the result of the deterministic relaxation. This normally takes about 10-20 iterations. The result of using learning in the deterministic relaxation is shown in Figure 1(d). About 10 learning cycles are used in this experiment. Figure 1(e) gives the result for the MPM algorithm after about 500 iterations. The boundary obtained by the MPM is the most accurate and also there are no misclassifications inside the homogeneous regions.

Example 2: (Grass, Raffia and Wood) Figure 2(a) shows this image and Figure 2(b) gives the coarse clustering obtained using $\rho = 0.3$. Note the presence of an ambiguous region (dark region at the top), which could not be classified into any of the other classes. The results of the various algorithms are shown in Figure 2(c)-2(e). Here again the MPM gave the best result.

5 Comments on an adaptive segmentation algorithm

In [1], a simpler model based on GRF is used to model the intensity process. This model can be summarized as:

$$Y_{ij} = X_{ij} + W_{ij} \tag{14}$$

where Y_{ij} is the observed intensity at location (i, j), X_{ij} is the true intensity and W_{ij} is an independent identically distributed zero mean gaussian noise and it is assumed that its variance is known. Further, $X_{ij} \in \{s_1, \ldots, s_N\}$, s_i being the intensity of the i-th region, and N are assumed to be known. The process X is modeled as a MRF taking one of these N values. The joint distribution of X can be written as a Gibbs distribution and the particular form of this used in [1] is called a Multilevel Logistic (MLL) distribution. Hence the parameters correspond to this MLL distribution. A maximum likelihood estimate of the parameters of the MLL distribution are computed and combined with simulated annealing to obtain an optimum segmentation of the scene. A convergence result is also proved for this adaptive segmentation scheme.

In the analysis of the algorithm, the assumptions made play an important role. Even with these simple assumptions, due to computational difficulties further approximations have to be made. For example in the above scheme, a pseudo-likelihood algorithm is used to approximate the MLEs to avoid the computational burden involved in the estimation of MLE. The use of simulated annealing makes the algorithm computationally demanding. Further, if any of the assumptions made above (eg., known number of regions, their intensity values or known noise parameters) are relaxed [12], the resulting convergence may not be even to a local optimum. Thus, even though the principle of simultaneous parameter estimation and segmentation could be used in more general cases like the textured images considered in this paper, it is not clear if it has any advantages compared to the scheme detailed in this paper where we first estimate the parameters from windows to obtain a coarse segmentation and then use pixel based schemes for finer segmentation.

5.1 A simple nearest-neighbor classification scheme

Adaptive segmentation should be data driven, but at the same time we should make use of whatever information that is available about the data in the design of such algorithms. For example in this paper we made an assumption that the texture intensities could be modeled by GMRF which simplified the parameter estimation significantly. To further illustrate the usefulness of the prior knowledge about data, we give below a simple classification scheme which makes the same assumptions as in the adaptive segmentation scheme of [1] and does not need expensive algorithms like simulated annealing to obtain comparable results. The data is the same as the one used in [1]. Also the information about the noise variance is not used in this segmentation operation. For obtaining the segmented image from a noisy version of it, we used the following algorithm:

1. At each pixel site (i, j), compute the average μ_{ij} in a small window (of size 3x3 in our case) around the pixel (i, j)

2. Then the intensity of the pixel is estimated as:

$$\hat{x}_{ij} = s_k = \min_l |s_l - \mu_{ij}| \tag{15}$$

3. Smooth the resulting texture by using a smoothing filter (similar to the one described in the previous section).

Figure 3 shows the performance of this scheme on a two region hand drawn image. Figure 3(a) is the original image with the two intensity levels being 100 and 150. This is one of the images used in [1]. Figure 3(b) is the noisy version with the noise being additive i.i.d. zero mean gaussian with standard deviation 25 (Signal to noise ratio (SNR) of 2). The classification result we obtained is shown in Figure 3(c) with the classification error of 1.73%. Figures 3(d) and 3(e) show the results when the noise deviation is 50 (SNR 1).

Corresponding results for the four region case (with intensity values 100,150,200 and 250) are shown in Figure 4. The maximum difference in the performance of the nearest

Algorithm	2 Regions		4 Regions	
	SNR 2	SNR 1	SNR 2	SNR 1
Adaptive Segmentation from [1]	0.96	3.88	0.40	1.98
Simple classification scheme described here	1.73	4.60	2.21	3.19

Table 1: Comparison of the adaptive segmentation algorithm in [1] with the nearest neighbor classification scheme. The numbers indicate percentage classification error. SNR 2 corresponds to a noise standard deviation of 25 and SNR 1 corresponds to a deviation of 50.

neighbor classification rule to that reported in [1] is for the four region case with SNR 2, where we obtained an error of 2.21% compared to 0.4% reported in [1]. Table 1 compares the performance of this nearest neighbor classification scheme with the adaptive segmentation algorithm of [1]. As far as the computation time required, this clustering technique takes few seconds of CPU time (for the 128x128 images, on a SUN-3 workstation) compared to 15-30 minutes (on VAX 8600) reported in [1].

As can be seen from these experiments, complexity of the segmentation algorithms can be greatly reduced by a proper use of prior information about the assumed models. The texture model considered in section 2 is more complicated than the one discussed here and the only explicit assumption made was that the textures can be modeled by a second order GMRF. Depending on the textures, this may or may not be a valid assumption. However from our experience with different real textures like wood, wool, water etc., this appears to be a good approximation and our experimental results also support this fact. We also find it better to separate the estimation stage from the segmentation stage. However, by separating estimation and segmentation, the algorithm will not lend itself to easy analysis.

6 Conclusions

Unsupervised segmentation is a difficult problem. Even before estimating the parameters of any assumed model, one has to decide whether the model is applicable to a particular image or not. As we observed in the previous section, sometimes the choice of appropriate models play a significant role. We feel that separating estimation stage from segmentation simplifies the problem and enables computationally manageable algorithms. In cases where the number of textures in an image is reasonably small, we are able to estimate the model parameters and segment the scene. We also noticed that by introducing a penalty for the small regions, which is equivalent to doing simple smoothing operations, deterministic relaxation schemes give results comparable to those of stochastic techniques like simulated annealing and MPM.

Acknowledgments

We would like to thank Prof. Derin of Univ. of Massachussets for providing us with the hand-drawn images (Figures 3a and 4a) used in section 5.

References

[1] S. Lakshmanan and H. Derin, "Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing", *IEEE Trans. on Pattern Anal. Machine Intell.*, vol. PAMI-11, pp. 799–813, August 1989.

[2] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random fields", *IEEE Trans. on Pattern Anal. Machine Intell.*, PAMI-9, pp. 39–55, January 1987.

[3] B. S. Manjunath, T. Simchony, and R. Chellappa, "Stochastic and deterministic networks for texture segmentation", *IEEE Trans. Acoust., Speech, Signal Process.*, pp. 1039–1049, June 1990.

[4] D. Geman, S. Geman, C. Graffigne, and P. Dong, "Boundary detection by constrained optimaization", pre-print, April 1989.

[5] F.S. Cohen and Z. Fan, "Unsupervised Textured Image Segmentation", Technical Report 86-1, Dept. of Elec Engg., Univ. of Rhode Island, June 1986.

[6] F.S. Cohen and D.B. Cooper, "Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian fields", *IEEE Trans. on Pattern Anal. Machine Intell.*, PAMI-9, pp. 195–219, March 1987.

[7] R. Chellappa, " Two-dimensional discrete Gaussian Markov random field models for image processing ", In *Progress in Pattern Recognition 2*, pp. 79–112, Ed. L.N. Kanal and A. Rosenfeld, Elsevier Science Publishers, North-Holland, 1985.

[8] R.L. Kashyap and R. Chellappa, "Estimation and choice of neighbors in spatial interaction models of images", *IEEE Transactions on Information theory*, IT-29, pp. 60–72, 1983.

[9] T. Simchony, R. Chellappa, and Z. Lichtenstein, "Relaxation algortihms for MAP estimation of grey level images with multiplicative noise", *IEEE Transactions on Information theory*, IT-36, pp. 608–613, May 1990.

[10] J. Besag, "On the statistical analysis of dirty pictures", *Journal of Royal Statistic Society B*, vol. 48, pp. 259–302, 1986.

[11] J.L. Marroquin, *"Probabilistic solution of inverse problems"*, PhD thesis, M.I.T, Artificial Intelligence Laboratory, September 1985.

[12] C. S. Won and H. Derin, "Unsupervised image segmentation using Markov random fields - part I: Noisy images", preprint.

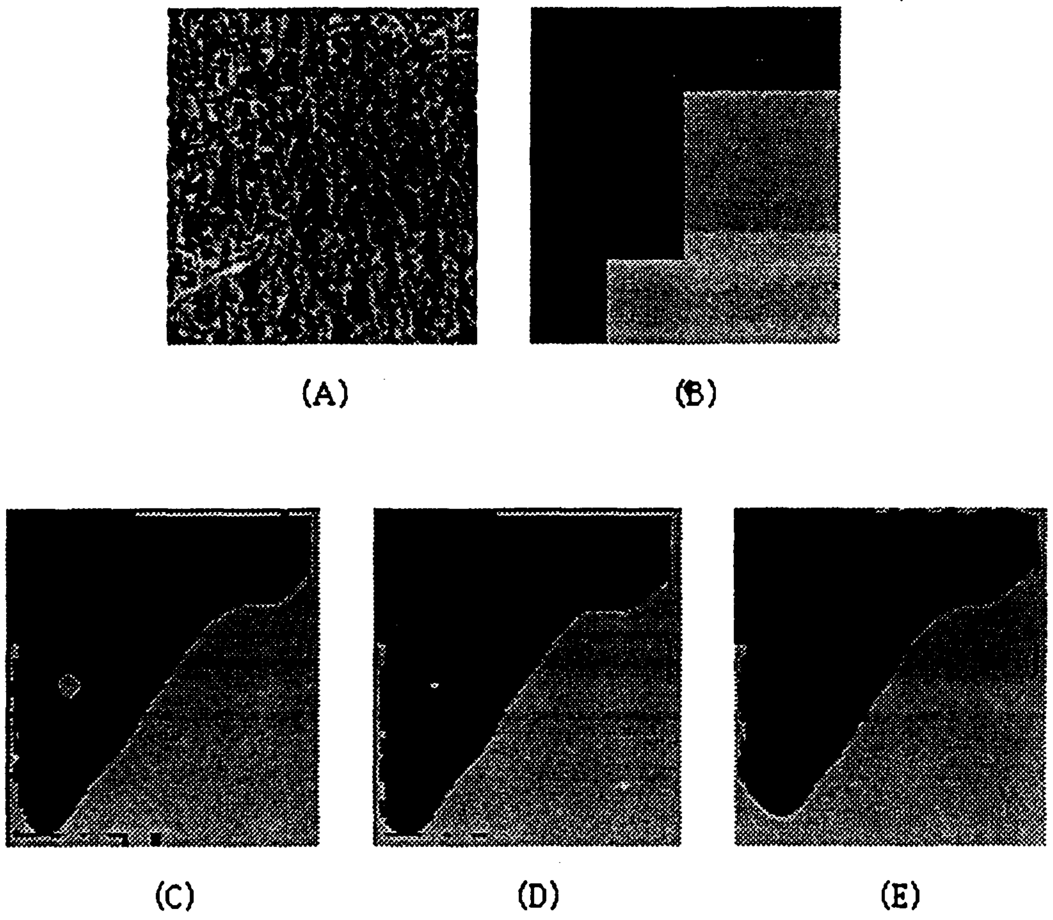


Figure 1: Unsupervised segmentation of an image consisting of two textures (grass and leather) (a) Original image (b) Coarse clustering (c) Deterministic relaxation (d) with learning (e) MPM result

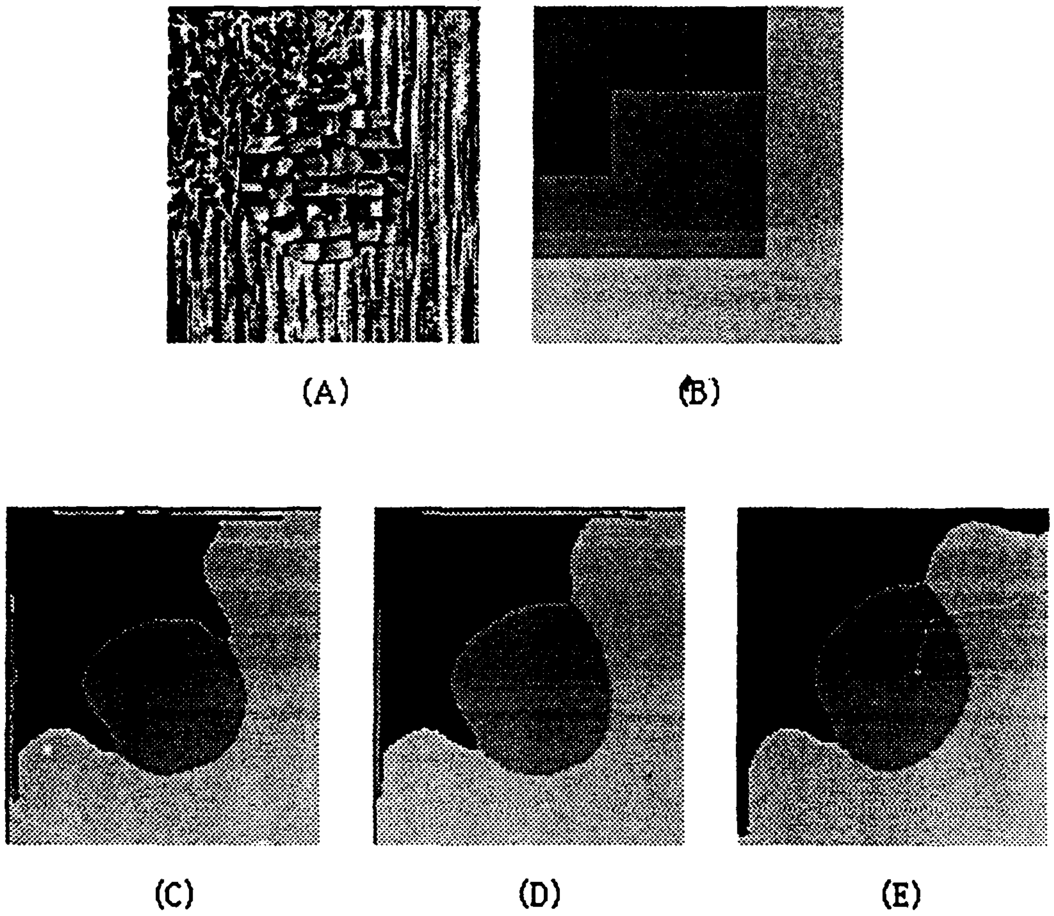


Figure 2: Unsupervised segmentation of an image having three textures (Grass, Raffia and Wood). (a) Original image (b) Coarse clustering. Note the presence of an ambiguous region (darkest region at the top) (c) Deterministic relaxation (d) with learning (e) MPM result

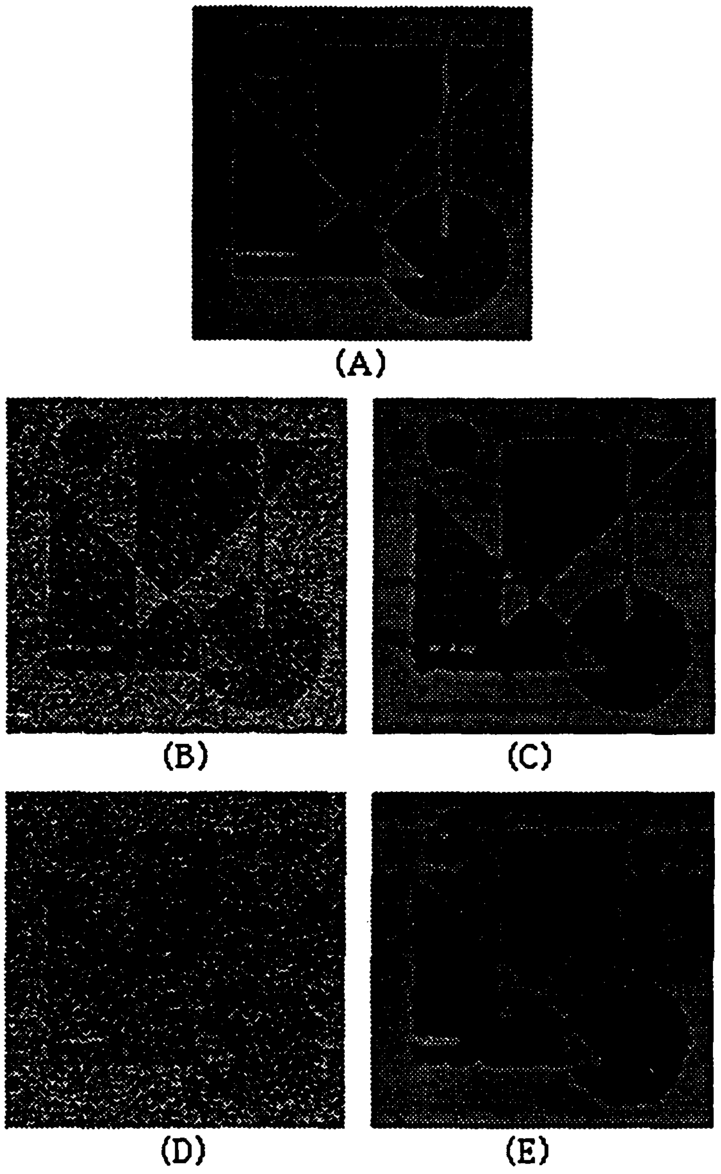


Figure 3: Segmentation of two region hand drawn image using a nearest neighbor classification rule. (a) original image. This is the same as the one used in [1]. (b) with SNR 2 (standard deviation 25) (c) segmented image from (b), (d) with SNR 1 (standard deviation 50) (e) segmented image from (d).

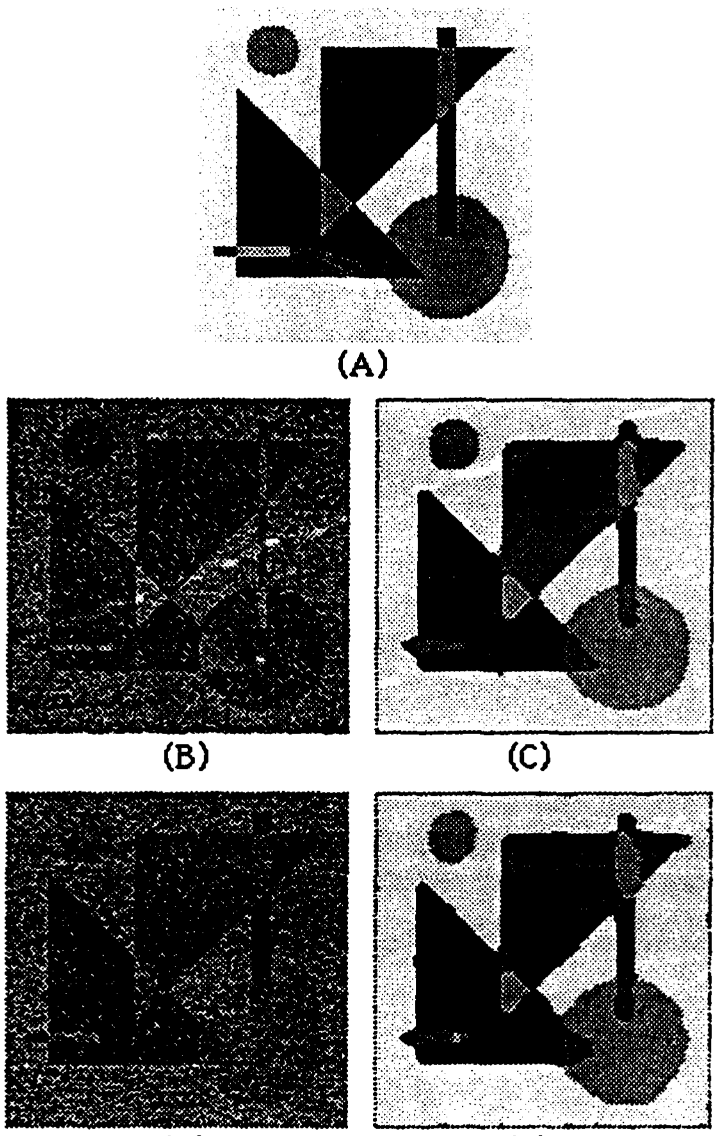


Figure 4: Segmentation of a four region hand drawn image using a nearest neighbor classification algorithm. (a) original image, same as the one used in [1]. (b) with SNR 2 (standard deviation 25) (c) segmented image from (b), (d) with SNR 1 (standard deviation 50) (e) segmented image from (d).

Neural Network Algorithms for Motion Stereo [1]

Y. T. Zhou and R. Chellappa

Signal and Image Processing Institute
Department of EE-Systems
University of Southern California

Abstract

Motion stereo method infers depth information from a sequence of image frames. Both batch and recursive neural network algorithms for motion stereo are presented. A discrete neural network is used for representing the disparity field.

The batch algorithm first integrates information from all images by embedding them into the bias inputs of the network. Matching is then carried out by neuron evaluation. This algorithm implements the matching procedure only once unlike conventional batch methods requiring matching many times.

Existing recursive approaches use either a Kalman filter or recursive least square algorithm to update the disparity values. Due to the unmeasurable estimation error, the estimated disparity values at each recursion are unreliable, yielding a noisy disparity field. Instead, our method uses a recursive least square algorithm to update the bias inputs of the network. The disparity values are uniquely determined by the neuron states after matching. Since the neural network can be run in parallel and the bias input updating scheme can be executed on line, a real time vision system employing such an algorithm is very attractived. A detection algorithm for locating occluding pixels is also included. Experimental results using natural image sequences are given.

1 Introduction

Motion stereo is a method for deriving depth information from either a moving camera or objects moving through a stationary 3-D environment. Since motion stereo uses more than two image frames, it usually gives more accurate depth measurements than static stereo which uses only two image frames. Applications of motion stereo are in ALV project and industrial robot vision systems. In this paper we present two neural network algorithms, batch and recursive, for computing disparities using a sequence of image frames based on the first order intensity derivatives and chamfer distance values. The chamfer distance value is defined as the distance from the non-edge pixel to the nearest edge pixel [1].

A substantial amount of work has been devoted to methods for computing the disparity field based on a sequence of images. In a relatively early paper, Nevatia [2] uses multiple views between two stereo views to achieve certain accuracy without an increase in search time. The object is placed on a turntable and multiple views are taken by a camera every 0.5 degree apart. Two simple methods are suggested for region search. Both methods only reduce the search range but not increase the resolution. Also they do not make use of any information acquired in the previous view for the next search procedure. Williams [3] presents a new approach for deriving depth from a moving camera in. By moving the camera forward, the disparity is estimated using simple triangulation. For simplicity, all the object surfaces are assumed to be flat and oriented in either horizontal direction i.e. parallel to image plane or vertical direction i.e. parallel to ground plane. Therefore, only the distances for horizontal surface and the height for vertical surface need to be found. To achieve subpixel accuracy, an image is interpolated according to the predicted disparity values obtained by a search process and occlusion effects. Based on the error between real and interpolated images, the correct orientation of each surface can be detected, and hence a segmented image consisting of refined synthetic surfaces can be obtained. For implementation purposes, an iterative segmentation procedure is employed and the systematic changes of distance and height embodied in synthetic segmented image at each iteration are used for finding the correct distance and height. Experimental results demonstrate the usefulness of this approach for simple natural image sequences. Since only planar surfaces with one of two different orientations are assumed to exist in natural images, areas corresponding to either non-planar surfaces or planar surfaces with other orientations are not correctly interpolated and therefore the estimated distances and heights for these areas are not reliable. Furthermore, this approach requires information about the focus of expansion (FOE) and the final result depends very much on the quality of initial segmentation.

Instead of computing depth in image space, Jain, Bartlett and O'Brien [4] developed a method for estimating the depth of feature points (corners) in the ego-motion complex logarithmic mapping (ECLM) space. They showed that the axial movement of the camera causes only horizontal but not the vertical change in the mapping of image points. Therefore, the depth of a feature point can be determined from the horizontal displacement in the ECLM for that point and from the camera velocity in the gaze direction. However, the mapping is very sensitive to noise, spatial quantization error and image blur, requiring some heuristics to establish the correspondence of points, such as thresholds

[1]This research work is partially supported by the AFOSR Contract No. F-49620-87-C-0007 and the AFOSR Grant No. 86-0196.

for maximum possible changes in the vertical direction and an upper bound for the search range in the horizontal direction in the ECLM space. Also the FOE for arbitrary translation of the camera and the feature points (corners) are assumed to be known. Another motion stereo method using feature points (corners) for computing depth in image space can be found in [5].

Recently, Xu, Tsuji and Asada [6] have suggested a coarse-to-fine iterative matching method for motion stereo. By sliding a camera along a straight line, a sequence of images is taken at predetermined positions. The pair with the short baseline is matched first to produce a coarse disparity map based on the zero-crossings. Then the coarse disparity map is used to reduce search range for the pair with the next longer baseline. This procedure is continued until the pair with the largest baseline is processed. One major advantage of this method is that occlusions can be predicted from the previous disparity map to avoid mismatches at present step. Although the computation time is less compared to other coarse-to-fine methods, this method gives only a sparse disparity map and can not be implemented on line.

Matthies, Szeliski and Kanade [7] have introduced two real time approaches, based on intensity values and features using a Kalman filtering technique. A sequence of lateral motion images is generated by a moving camera along a straight line from left to right (or right to left). The intensity based approach consists of four stages for each frame. First, a new measurement of disparity at each pixel is obtained by using a correlation matching procedure. Then the estimate of disparity is updated using a Kalman filter update equation based on the new measurement. Third, a generalized piecewise continuous spline technique is used to smooth the updated estimate. Finally, the disparity for each pixel in the next frame is given by the prediction procedure. As reported in [7], the intensity based approach is more efficient than the feature based approach. But a major problem in the intensity based approach is that once the updated estimate is smoothed in the third stage, the gain of the Kalman filter and the error variance of the estimate are no longer correct so that they can not be used in the next iteration.

Attempting to achieve human-like performance, many researchers have been using neural networks to solve the matching problem based on binocular images i.e. one pair of images [8, 9, 10, 11, 12, 13, 14, 15]. In this paper, we first present a neural network based batch algorithm for motion stereo. The conventional batch algorithms implement matching procedure many times requiring a lot of computations. For example, if there are M image frames, the matching procedure has to be implemented $(M-1)$ times to obtain $(M-1)$ disparity measurements for each pixel. To obtain a good estimate of disparity value from these measurements, usually a filtering procedure is required. Instead of doing matching $(M-1)$ times, the neural network batch approach implements the matching algorithm only once by simultaneously using all the images pairs so that computational complexity is greatly reduced. We also present a real time algorithm using a neural network. Basically, we use a recursive least squares (RLS) algorithm to update the bias inputs of the network whenever the next frame becomes available. After all images are received, matching is carried out by neuron evaluation, minimizing energy function of the network. The disparity values are then given by the neuron states. If the intermediate results are required, one can implement the matching procedure for every pair of images. Unlike [7], this method runs the matching algorithm only once and needs no interpolation procedure. Since the neural network can be run in parallel and the RLS algorithm can be implemented on line, the recursive algorithm is extremely fast and hence useful for real time robot vision applications. As the derivatives of the intensity function are more reliable than the intensity values and are dense, both algorithms use the derivatives as measurement primitives for matching. The chamfer distance information is also used for matching to overcome the lack of information in homogeneous regions. Recognizing that occlusions might affect the matching accuracy very seriously when a long sequence of image frames is involved, we have designed an algorithm for locating occluding pixels. Once an occluding pixel is detected, the occlusion information is embedded into the network by resetting the bias inputs so that the network automatically takes care of occluding pixels during the updating procedure.

2 Depth from Motion

2.1 Camera Configuration

It is assumed that a sequence of images is taken by a camera moving along with a constant velocity from right to left along a straight line, as shown in Figure 1. Several assumptions are made for simplifying the problem. First, it is assumed that the optical axis of the camera is perpendicular to the moving direction and the horizontal axes of the image planes are parallel

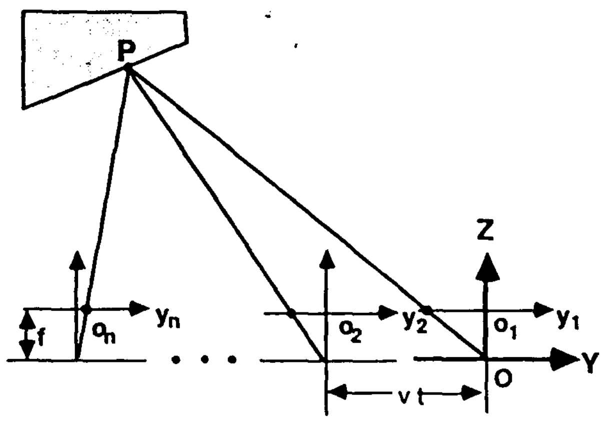


Figure 1: Camera geometry for motion stereo.

to the moving direction. The constraint imposed on the camera configuration is to restrict the search within the horizontal direction only, the so called epipolar constraint. Secondly, it is assumed that the camera takes pictures exactly every t seconds apart. Thus, all images are equally separated, i.e. each successive image pairs has the same baseline.

Let $OXYZ$ be the world coordinate system with Z axis directing along the camera optical axis and $o_i x_i y_i$ be the pth image plane coordinate system. The origin of the pth image system is located at $(0, -(p-1)vt, f)$ of the world system, where v is the velocity of camera, vt is the distance between two successive images and f is the focal length of the lens which takes a positive value in the world system. Under perspective projection, a point in the world, (X_o, Y_o, Z_o), projects into the pth image plane at

$$(x_i, y_i) = (\frac{f X_0}{Z_o}, \frac{f(Y_o + (p-1)vt)}{Z_o}) \tag{1}$$

Theoretically, the disparity D_o can be derived from two successive image frames

$$D_o = y_p - y_{p-1} = f\, d\, \frac{1}{Z_o} \tag{2}$$

where $d = v\,t$ is the baseline.

2.2 Estimation of Pixel Positions

Let (i,j) be the position of the (i,j)th pixel in the first frame. In the successive frames, due to camera motion, the position of all pixels are shifted to the right by vt. Under the epipolar constraint, the shift happens only in the horizontal direction. For example, the (i,j)th pixel moves from position (i,j) in the first frame to position $(i, j+\frac{fvt}{Z_{i,j}})$ in the second frame. Let $S_{i,j}(p)$ be the total shift of pixel (i,j) from the first to the pth frame. Thus

$$\begin{aligned} S_{i,j}(p) &= (p-1)\frac{fvt}{Z_{i,j}} \\ &= (p-1)\, d_{i,j} \end{aligned} \tag{3}$$

where $d_{i,j}$ is the true disparity value for pixel (i,j). Note that the shift $S_{i,j}(p)$ is continuous due to the continuous variable $d_{i,j}$. A rounding operation has to be applied to $S_{i,j}(p)$ for locating the (i,j)th pixel in the subsampled image. After rounding, the position of the (i,j)th pixel in the pth frame is given by

$$(i, j + [\frac{S_{i,j}(p)}{W}]W). \tag{4}$$

where [ ] is a rounding operator. It can be simply written as

$$(i, j + kW) \tag{5}$$

where

$$k = [\frac{S_{i,j}(p)}{W}].$$

3 Matching Algorithms

Two algorithms, batch and recursive, are presented in this section.

3.1 Batch Algorithm

A discrete neural network is used for representing the disparity field. The network consists of $N_r \times N_c \times (D+1)$ mutually interconnected binary neurons, where D is the maximum disparity, N_r and N_c are the image row and column sizes, respectively. Let $V = \{v_{i,j,k}, 1 \le i \le N_r, 1 \le j \le N_c, 0 \le k \le D\}$ be a binary state set of the neural network with $v_{i,j,k}$ (1 for firing and 0 for resting) denoting the state of the (i,j,k)th neuron. For each pixel, say (i,j), we use $(D+1)$ mutually exclusive neurons $\{v_{i,j,0}, v_{i,j,1}, ..., v_{i,j,D}\}$ to represent the disparity value. Theoretically, disparity takes continuous values. For implementation purposes, we sample the disparity range using bins of size W. Hence, when $v_{i,j,k}$ is 1, this means that the disparity value is kW at the pixel (i,j).

The neural network parameters, the interconnection strengths $T_{i,j,k;l,m,n}$ and the bias inputs $I_{i,j,k}$, can be determined in terms of the energy function of the network. The energy function of the neural network is defined as

$$\begin{aligned} E = & -\frac{1}{2}\sum_{i=1}^{N_r}\sum_{l=1}^{N_r}\sum_{j=1}^{N_c}\sum_{m=1}^{N_c}\sum_{k=0}^{D}\sum_{n=0}^{D} T_{i,j,k;l,m,n}\, v_{i,j,k}\, v_{l,m,n} \\ & -\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=0}^{D} I_{i,j,k}\, v_{i,j,k} \end{aligned} \tag{6}$$

In order to use the spontaneous energy-minimization process of the neural network, we reformulate the stereo matching problem under the epipolar assumption as one of minimizing an error function with constraints. Suppose that M image frames are used for matching, the error function can be written as

$$\begin{aligned} E = & \frac{1}{M-1}\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=0}^{D}\sum_{p=1}^{M-1} [(g'_p(i, j+(p-1)kW) \\ & -g'_{p+1}(i, j+pkW))^2 + \kappa(f_p(i, j+(p-1)kW) \\ & -f_{p+1}(i, j+pkW))^2]\, v_{i,j,k} \\ & +\frac{\lambda}{2}\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=0}^{D}\sum_{s\in S} (v_{i,j,k} - v_{(i,j)+s,k})^2 \end{aligned} \tag{7}$$

where $\{g'_p(\cdot)\}$ and $\{f_p(\cdot)\}$ denote the intensity derivatives and the chamfer distance values at $(\cdot)$ of the pth frame, respectively, S is an index set excluding (0,0) for all the neighbors in a $\Gamma \times \Gamma$ window centered at point (i,j), λ and κ are constants. The first term called the photometric constraint in (7) seeks disparity values such that all regions of two images are matched in a least squares sense. Meanwhile, the second term is the smoothness constraint on the solution. The constant κ determine the relative importance of the two kinds of measurement primitives, derivatives and distance values, and the constant λ determines the tradeoff between the two terms to achieve the best results.

The interconnection strengths and bias inputs are determined by comparing the terms in the expansion of (7) with the corresponding terms in (6)

$$T_{i,j,k;l,m,n} = -48\lambda\delta_{i,l}\delta_{j,m}\delta_{k,n} + 2\lambda\sum_{s\in S}\delta_{(i,j),(l,m)+s}\delta_{k,n} \tag{8}$$

and

$$\begin{aligned} I_{i,j,k} = & -\frac{1}{M-1}\sum_{p=1}^{M-1} [(g'_p(i, j+(p-1)kW) \\ & -g'_{p+1}(i, j+pkW))^2 + \kappa(f_p(i, j+(p-1)kW) \\ & -f_{p+1}(i, j+pkW))^2] \end{aligned} \tag{9}$$

where $\delta_{a,b}$ is the Dirac delta function. The size of the smoothing window used in (8) is 5. However, one can choose either a larger or smaller window. From (8) one can see that the interconnections are symmetric, the self-feedback weight $T_{i,j,k;i,j,k}$ is not zero. Also note that the bias inputs consists of the measurement primitives only and the interconnection strengths contains no information about the images.

Matching is carried out by neuron evaluation. Once the parameters $T_{i,j,k;l,m,n}$ and $I_{i,j,k}$ are obtained using (8) and (9), each neuron can synchronously evaluate its state and readjust according to updating equations

$$u_{i,j,k} = \sum_{l=1}^{N_r} \sum_{m=1}^{N_c} \sum_{n=0}^{D} T_{i,j,k;l,m,n} v_{l,m,n} + I_{i,j,k} \qquad (10)$$

and

$$v_{i,j,k} = g(u_{i,j,k}) \qquad (11)$$

where $g(x_{i,j,k})$ is a *maximum* evolution function

$$g(x_{i,j,k}) = \begin{cases} 1 & if \;\; x_{i,j,k} = max(x_{i,j,l}; l = 0,1,...,D). \\ 0 & otherwise. \end{cases} \qquad (12)$$

Note that the synchronous updating scheme can be implemented in parallel. The uniqueness of matching problem is ensured by a batch updating scheme–$D+1$ neurons $\{v_{i,j,0}, ...v_{i,j,D}\}$ at pixel (i,j) are updated at each step simultaneously.

The initial state of the neurons were set as

$$v_{i,j,k} = \begin{cases} 1 & if \;\; I_{i,j,k} = max(I_{i,j,l}; l = 0,1,...,D). \\ 0 & otherwise \end{cases} \qquad (13)$$

where $I_{i,j,k}$ is the bias input.

As motioned in [16, 15], the self-feedback may cause energy function to increase with a transition. A deterministic decision rule is used to ensure convergence of the network, probably to a local minimum. The batch matching algorithm can then be summarized as

Batch matching algorithm:

1. Estimate the network inputs.
2. Set the initial state of the neurons.
3. Update the state of all neurons synchronously according to the deterministic decision rule.
4. Check the energy function; if energy does not change anymore, stop; otherwise, go back to step 3.

3.2 Recursive Algorithm

This algorithm basically consists of two steps: bias input update and stereo matching. Whenever a new frame of image becomes available, the bias inputs of the network are updated by the RLS algorithm.

Suppose that images are corrupted by additive white noise and the measurement model is given by

$$\begin{aligned} \bar{I}_{i,j,k}(p) &= h(p, g'_p(i,j+(p-1)kW), f_p(i,j+(p-1)kW), \\ & \quad g'_{p+1}(i,j+pkW), f_{p+1}(i,j+pkW), n_{i,j,k}(p)) \\ &= -(\bar{g}'_p(i,j+(p-1)kW) - \bar{g}'_{p+1}(i,j+pkW))^2 \\ & \quad -\kappa(\bar{f}_p(i,j+(p-1)kW) - \bar{f}_{p+1}(i,j+pkW))^2 \\ & \quad for \quad p = 1,2,...,M-1 \end{aligned}$$

where h is a measurement function and $n_{i,j,k}(p)$ is noise. For p such measurements, find a function

$$\hat{I}_{i,j,k}(p) = \hat{I}_{i,j,k}(p, \bar{I}_{i,j,k}(p), \bar{I}_{i,j,k}(p-1), ..., \bar{I}_{i,j,k}(1)) \qquad (14)$$

that estimates the value of the bias input $I_{i,j,k}$ in some sense. The value of the function is the estimate. If the measurement function is linear and the measurement noise is white, then a Kalman filter is commonly used for finding an optimal estimate. In (14), as the measurement function is nonlinear and the measurement noise is no longer white but is dependent on measurements, the linear Kalman filter does not yield a good estimate. In contrast to the Kalman filter, the RLS algorithm does not make any assumption about measurement function and noise. Hence, the RLS algorithm can be used to update the bias inputs. When the pth frame becomes available, the bias input is updated by

$$I_{i,j,k}(p) = I_{i,j,k}(p-1) + \frac{1}{p}(\bar{I}_{i,j,k}(p) - I_{i,j,k}(p-1)), \qquad (15)$$

This RLS algorithm is equivalent to the batch least squares algorithm with the initial condition

$$I_{i,j,k}(0) = 0.$$

The interconnection strengths is the same as in (8) and the bias input is given by (15). Since the bias inputs are recursively updated and contain all the information about the previous images, we do not have to implement the matching algorithm for every recursion if the intermediate result are not required. This method greatly reduces the computational load and therefore is extremely fast. Formally, the algorithm is as follows:

Recursive matching algorithm:

1. Update the bias inputs using the RLS algorithm.
2. Initialize the neuron states.
3. If there is a new frame coming, go back to step 1; otherwise go to step 4.
4. Update the neuron states using (10) and (11) using the deterministic decision rule.

This algorithm has several advantages over the correlation algorithm of [7]:

1. This algorithm recursively updates the bias inputs instead of the disparity values. The matching algorithm is implemented only once.
2. This algorithm incorporates the smoothness constraint into the matching procedure instead of using an extra smoothing procedure.
3. This algorithm uses the derivatives of the intensity func-

tion, which are more reliable than the intensity values, as measurement primitives. Hence it is suitable for natural images.

4 Estimation of Measurement Primitives

In this section we present two methods for estimation of the derivatives and chamfer distance values.

4.1 Estimation of Derivatives

As only derivatives in the horizontal direction are required for matching, the epipolar constraint saves a lot of computations. By using a set of univariate discrete Chebyshev polynomials to approximate the intensity function in a window, we have

$$\hat{g}(i,j+y) = \mathbf{A}^t\, \underline{\mathrm{CH}}(y) \tag{16}$$

where $\hat{g}(i,j+y)$ is the approximated continuous intensity function, t denotes the transpose operator,

$$\mathbf{A}^t = [a_0, a_1, a_2, a_3, a_4] \tag{17}$$

is the coefficient vector and

$$\underline{\mathrm{CH}}^t(y) = [Ch_0(y),\ Ch_1(y),\ Ch_2(y),\ Ch_3(y),\ Ch_4(y)] \tag{18}$$

is polynomial vector defined over an index set $\Omega = \{-\omega, -\omega+1, ..., \omega-1, \omega\}$, i.e. over a window of size $2\omega+1$, as in [15]. The coefficients $\{a_n\}$ are estimated using

$$a_n = \frac{\sum_{y'\in\Omega} Ch_n(y')\, g(i,j+y')}{\sum_{u\in\Omega} Ch_n^2(u)} \tag{19}$$

where $\{g(i,j+y')\}$ are the observed intensity values.

The first order derivatives of the intensity function at subpixel position $(i,j+y)$ can be calculated by

$$\frac{\partial g(i,j)}{\partial j}\Big|_{j=j+y} = \frac{d\hat{g}(i,j+y)}{dy} = \mathbf{A}^t \frac{d}{dy}\underline{\mathrm{CH}}(y) \tag{20}$$

$$for \quad -0.5 \le y < 0.5$$

For simplicity of notation, we use $g'(i,j+y)$ to represent the first order partial derivatives of the subpixel intensity function. Using (17), (18) and (19) in (20) we get

$$g'(i,j+y) = \sum_{y'\in\Omega} \underbrace{(\sum_{l=0}^{4} \frac{Ch_l(y')Ch_l(y)}{\sum_{u\in\Omega} Ch_l^2(u)})}_{W(y,y')} g(i,j+y'). \tag{21}$$

In (21), $W(y,y')$ can be considered as a window operator. Hence, the derivatives can be obtained by convolving a window $W(y,y')$ with the input image. The window size can be determined by properly considering the effects of image noise and spatial quantization error [16].

4.2 Estimation of Chamfer Distance Values

To estimate the distance values, two steps are involved: converting the intensity image into the binary image consisting of the edge and non-edge pixels and transforming the binary image to the chamfer image.

Many conventional edge detectors can be used to find edge pixels from the intensity image. Since matching is restricted in the horizontal direction, only the horizontal chamfer edge information is needed and therefore only the vertical edges have to be detected. For simplicity of implementation, we use the Prewitt edge detector [17] with window size of 3×3 for detecting the vertical edges.

The chamfer distance values are iteratively determined using the following algorithm

$$f_{i,j}^{(l)} = min(f_{i,j-1}^{(l-1)} + 2, f_{i,j}^{(l-1)}, f_{i,j+1}^{(l-1)} + 2) \tag{22}$$

where $f_{i,j}^{(l)}$ is the distance value at the (i,j)th pixel and the l denotes the iteration number. Initially, the distance values are set to zero for edge pixels and nonzero (say, 1000) for non-edge pixels. The edge pixels obviously get the value zero. This algorithm is completely parallel and the iteration number is equal to the longest distance value occurring in the image.

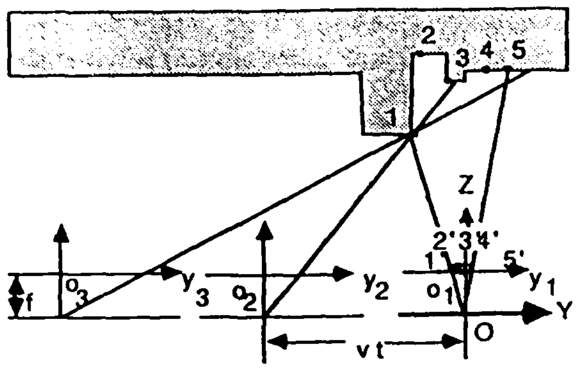
5 Detection of Occlusions

Detection of occluding pixels is an important issue in motion stereo. In this section we first discuss the nature of occlusion and then derive a mean matching error. Finally, a detection algorithm is given for locating occluding pixels based on the mean matching error.

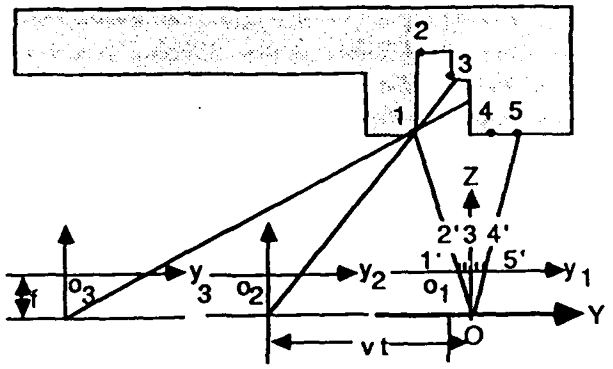
5.1 Occlusions

As shown in Figure 2(a), when a camera moves from right to left, points 2, 3, 4 and 5 project into the first image plane at 2', 3', 4' and 5'. But points 2 and 3 on the object surface will not project into the second image plane because they are occluded by the front surface on which point 1 lies. Similarly, points 2, 3, 4 and 5 will not appear in the image plane 3. At the location of pixels 2', 3', 4' and 5', the match error is usually large which means no conjugate pixels can be found in the successive image frames. Hence, the disparity values are undetermined. Such pixels are called occluding pixels. When a smoothness constraint is used, although the matching algorithm always assign some values to the occluding pixels, the discontinuities of the disparity field may be shifted. As the number of frames increases, the number of occluding pixels will dramatically increase too. For instance, if only two object points are occluded for the second image as shown in Figure 2(a), then about ten points are occluded for the sixth image which gives a ten pixel wide occluding region in the first image plane. On the other hand, if only the first two frames are used for matching, then pixels 4' and 5' are not occluding pixels and therefore the disparity values at such location are determinable. As the third frame does not provide any information about pixels 4' and 5', there is no need to update the bias inputs at the location of these pixels.

However, in some cases the number of occluding pixels does not increase as the number of frames increases. One typical example is illustrated in Figure 2(b), where pixels 4' and 5' are not occluding pixels when the third image frame is used.



(a) Pixels 2', 3', 4' and 5' are occluding pixels.



(b) Pixels 4' and 5' are not occluding pixels.

Figure 2: Occluding pixels.

5.2 Matching Error

When multiple frames are used for matching, the spatial quantization error usually causes a matching error. In this section, we derive a mean matching error which can be used to detect occluding pixels. It is assumed that the true disparity at pixel (i,j) in a smooth region can be expressed as

$$d_{i,j} = kW + \delta_{i,j}$$

where $\delta_{i,j}$ is uniformly distributed in $[-\frac{W}{2}, \frac{W}{2})$, and the first order derivative of the intensity function at point $(i, j + (p-1)kW)$ of the pth image can be expanded as a Taylor series about the point $(i, j + (p-1)(kW + \delta_{i,j}))$ as

$$\begin{aligned} g'(i,j+(p-1)kW) &= g'(i,j+(p-1)(kW+\delta_{i,j}) - \\ &\quad p\delta_{i,j}g''(i,j+(p-1)(kW+\delta_{i,j}) \\ &\quad -\delta_{i,j}g''(i,j+(p-1)(kW+\delta_{i,j}) \\ &\quad +O(\delta_{i,j}^2). \end{aligned}$$

$$for \quad p = 2, 3, ..., M \tag{23}$$

where $g''(\cdot)$ denotes the second order derivative. The best estimate of the disparity value is given by

$$\hat{d}_{i,j} = kW.$$

The matching error can be approximately written as

$$\begin{aligned} Error(i,j)_k &= \frac{1}{M-1}\sum_{p=1}^{M-1}[g_p'(i,j+(p-1)kW) \\ &\quad -g_{p+1}'(i,j+pkW)]^2 \\ &\simeq \frac{1}{M-1}\sum_{p=1}^{M-1}[g_p'(i,j+(p-1)(kW+\delta_{i,j})) \\ &\quad -g_{p+1}'(i,j+p(kW+\delta_{i,j})) \\ &\quad -(p-1)\delta_{i,j}g_p''(i,j+(p-1)(kW+\delta_{i,j})) \\ &\quad +p\,\delta_{i,j}g_{p+1}''(i,j+p(kW+\delta_{i,j}))]^2 \end{aligned} \tag{24}$$

When images are corrupted by an additive white noise variable with variance σ_n^2, the mean error at (i,j) becomes [18]

$$\mathrm{E}\{Error(i,j)_k\} = C_1\sigma_n^2 + C_2(g_1''(i,j))^2 \tag{25}$$

where C_1 and C_2 are determined by

$$\begin{aligned} C_1 &= \frac{1}{M-1}\sum_{p=1}^{M-1}\{\sum_{n=1}^{4}\frac{1}{\sum_{v\in\Omega}Ch_n^2(v)}[(\frac{d}{dy}Ch_n(y))^2|_{y=\bar{y}_{p-1}} \\ &\quad +(\frac{d}{dy}Ch_n(y))^2|_{y=\bar{y}_p}] \\ &\quad +\frac{W^2}{12}\sum_{n=2}^{4}[p^2(\frac{d^2}{dy^2}Ch_n(y))^2|_{y=\bar{y}_p} \\ &\quad +(p-1)^2(\frac{d^2}{dy^2}Ch_n(y))^2|_{y=\bar{y}_{p-1}}]\}, \end{aligned}$$

$$C_2 = \frac{W^2}{12},$$
$$\bar{y}_p = pkW - [pkW]$$

and [ ] is a rounding operator. If the variance of noise and the second order derivatives of the intensity function are known or estimated from the images, then the mean error corresponding to the disparity value k at every point for a given ω (window size), M (frame number) and W (width of subsample interval) can be calculated.

5.3 Detection of Occluding Pixels

The intuitive analysis given in Section 5.1 essentially suggests a method for detecting occluding pixels. By using the mean matching error derived above the following detection rule can be used for detecting occluding pixels and hence we can prevent the RLS algorithm from updating the bias input at the location of occluding pixels.

<u>Detection rule</u>: An occluding pixel at location (i,j) is detected if

$$\begin{aligned} &min(I_{i,j,k}|_{\kappa=0};\ 0 \le k \le D) > \\ &max(\mathrm{E}\{error(i,j)_k\};\ 0 \le k \le D) + b \end{aligned} \tag{26}$$

where $b \ge 0$ is a constant for raising the threshold. When the noise variance is unknown, one can use a constant threshold instead of the mean error.

For the recursive algorithm, once an occluding pixel is detected, the bias inputs of neurons at such locations will not be updated anymore. But during the first iteration, the bias inputs of neurons at the locations of occluding pixels are first updated and then corrected accordingly. The correction procedure is as follows. From Figure 2 it can be seen that the pixels on the left side of the occluding region have high disparity values and the pixels on the right side have low disparity values. The width of the occluding region, i.e. the number of the occluding pixels is approximately given by

$$\hat{\Delta}_{i,j} = [\frac{D_{i,j-1} - D_{i,j+\Delta_{i,j}}}{W}] \qquad (27)$$

where [ ] is a rounding operator, (i, j) denotes the location of the most left occluding pixel, $\Delta_{i,j}$ is the true width, $\hat{\Delta}_{i,j}$ is a estimate of the width, and $D_{i,j-1}$ and $D_{i,j+\Delta_{i,j}}$ are the disparity values of the nearest left and right nonoccluding pixels, respectively. Since a smoothness constraint is used, the occluding pixel usually takes either the high disparity value $D_{i,j-1}$ or low disparity value $D_{i,j+\Delta_{i,j}}$. The discontinuities of the disparity field can be detected by checking the disparity values in the y_1 direction if there is a transition from the high value to the low value. Starting with the discontinuity pixel, we check all the left and right neighbor pixels. The search procedure will not be stopped until a $\hat{\Delta}_{i,j}$ wide or less occluding region including the discontinuity pixel is found. Then, for all occluding pixels the bias input of the $\Delta_{i,j}$th neuron is corrected by

$$I_{i,j,D_{i,j+\Delta_{i,j}}}(1) = min(I_{i,j,k}(1);\ k = 0, 1, ..., D). \qquad (28)$$

For the batch algorithm, the bias inputs at occluding pixels are estimated using only the first two image frames.

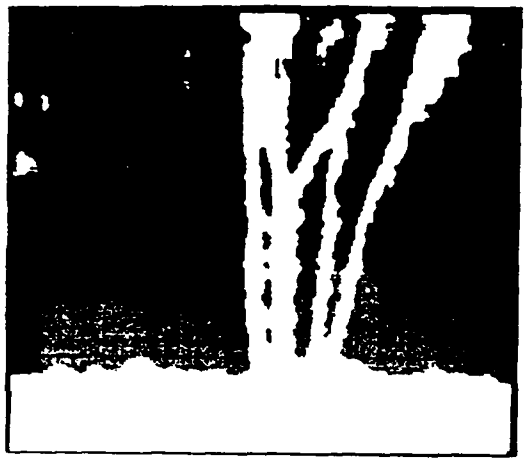
6 Experimental Results

We have tested both the batch and recursive algorithms on two sequences of natural images taken by a camera moving right to left. Due the space limitations, we give only one experimental result here. The images are of size 256 × 233. We arbitrarily chose five successive frames for testing, although there is no limit to the number of frames that can be used. Figure 3 (a) shows the first frame. No alignment in the vertical direction was made and the maximum disparity, about 2 pixels, was measured by hand. Same parameters were chosen for both algorithms. The subpixel width W was set at 0.2 and hence $D = 10$. The parameter λ and κ were set at 20 and 5, respectively. The threshold for occluding pixel detection was set at 150 because the noise variance is unknown. Figure 3 (b) shows the batch result after 45 iterations. The disparity map is represented as an intensity image with the brightest value denoting the maximum disparity value. The recursive result is shown in Figure 3 (c). The iterations for the recursive solution are 20. Since at occluding regions we still use the measurement primitives extracted from first two image frames for matching, the algorithms might generate some isolated points or regions (at most $\lfloor WD \rfloor$ pixels wide) due to the incorrect information caused by the occlusions. To remove such points and regions, a median filter is used in our experiments.

7 Discussion

We have presented two neural network based algorithms, known

(a) The Trees image.


(b) Batch algorithm.

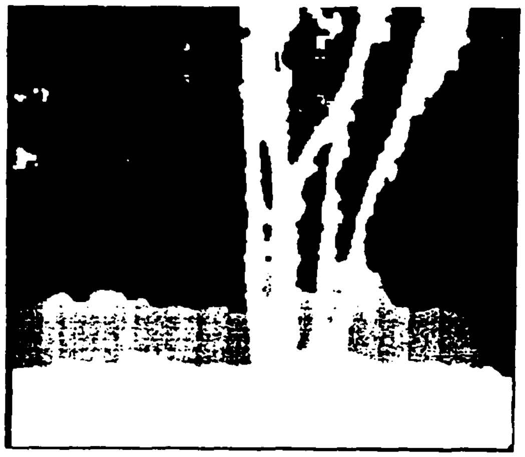

(c) Recursive algorithm.

Figure 3: Disparity maps.

as the batch and recursive algorithms, for motion stereo using the first order derivatives of intensity function as measurement primitives. For the recursive algorithm, the bias inputs of the neurons are recursively updated and if the intermediate results are not required the matching procedure is implemented only once. Unlike existing recursive algorithms, the disparity field obtained by our algorithm is smooth and dense. Also no batch results are needed for setting the initial states of the neurons. Both batch and recursive methods gave very good results in comparison to Barnard's approach [19]. Experimental results show that the recursive algorithm needs fewer iterations than the batch algorithm. This is because the recursive algorithm uses a better bias input updating scheme, especially for the occluding pixels. The good estimate of the bias inputs makes the network converge fast, although the updating step for bias inputs takes more computations. In view of parallelism and fast convergence, the recursive algorithm is useful for real time implementation, such as in a robot vision system. In our experiment, the threshold used was 150 which seems a little bit conservative. However, the maximum disparity is only about 2 pixels which means that the width of the occluding region is less than 2 pixels for two frames and there are only a few occluding pixels along the right boundaries of the trees. Hence the occluding pixels do not cause a serious problem in this experiment. This is also why the iteration number does not reduce a lot. We believe that if the maximum disparity is large and a long sequence of images is used, then the improvement on the occluding pixel detection will greatly reduce the number of iterations.

References

[1] H. G. Barrow, J. M. Tenenbaum, R.C. Bolles, and H. C. Wolf, "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching", In *Proc. Fifth International Joint Conf. on Artificial Intelligence*, Cambridge, MA, 1977.

[2] R. Nevatia, "Depth Measurement by Motion Stereo", *Computer Graph. and Image Processing*, vol. 6, pp. 619-630, 1976.

[3] T. Williams, "Depth from Camera Motion in a Real World Scene", *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. PAMI-2, pp. 511-516, November 1980.

[4] R. Jain, S. L. Bartlett, and N. O'Brien, "Motion Stereo Using Ego-Motion Complex Logarithmic Mapping", *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. PAMI-9, pp. 356-369, May 1987.

[5] H. Itoh, A Miyauchi, and S. Ozawa, "Distance Measuring Method Using Only Simple Vision Constrained for Moving Robots", In *Seventh Intl. Conf. Pattern Recognition*, pp. 192-195, Montreal, July 1984.

[6] G. Xu, S. Tsuji, and M. Asada, "A Motion Stereo Metho Based on Coarse-to-Fine Control Strategy", *IEEE Trans on Patt. Anal. and Mach. Intel.*, vol. PAMI-9, pp. 332-336, March 1987.

[7] L. Matthies, R. Szeliski, and T Kanade, "Kalman Filter Based Algorithms for Estimating Depth from Image Sequences", In *Proc. DARPA Image Understanding Work shop*, pp. 199-213, Cambridge, MA, April 1988.

[8] P.Dev, "Perception of Depth Surfaces in Random-dot Stereogram: a Neural Model", *Int. J. Man-Machine Studies*, 7, pp. 511-528, 1975.

[9] D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity", *Science*, vol. 194, pp. 283-287, October 1976.

[10] T. Poggio, "Vision by Man and Machine", *Scientific American*, vol. 250, pp. 106-116, April. 1984.

[11] N. M. Grzywacz and A. L.Yuille, "Motion Correspondence and Analog Networks", In *Proc. Conf. on Neural Networks for Computing*, pp. 200-205, Snowbird, UT, 1986, American Institute of Physics.

[12] C. V. Stewar and C. R. Dyer, "A Connectionist Model fo Stereo Vision", In *Proc. IEEE First Annual Intl. Conf. o Neural Networks*, San Diego, CA, June 1987.

[13] G. Z. Sun, H. H. Chen, and Y. C. Lee, "Learning Stereop with Neural Networks", In *Proc. IEEE First Annual In Conf. on Neural Networks*, San Diego, CA, June 1987.

[14] A. F. Gmitro and G. R. Gindi, "Optical Neurocompute Implementation of the Marr–Poggio Stereo Algorithm", *Proc. IEEE First Annual Intl. Conf. on Neural Netwo* San Diego, CA, June 1987.

[15] Y. T. Zhou and R. Chellappa, "Stereo Matching Using Neural Network", In *Proc. Intl. Conf. on Acoustics, Speec and Signal Processing*, pp. 940-943, New York, NY, Apr 1988.

[16] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Im age Restoration Using a Neural Network", *IEEE Tran Acoust,Speech,Signal Processing*, vol. 36, pp. 1141-1151 July 1988.

[17] J. M. Prewitt, "Object Enhancement and Extraction", In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processin and Psychopictorics*, Academic Press, New York, 1970.

[18] Y. T. Zhou, *"Artificial Neural Network Algorithms fo Some Computer Vision Problems"*, PhD thesis, Universit of Southern California, Los Angeles, CA, November 1988

[19] S. T. Barnard, "A Stochastic Approach to Stereo Vision", In *Proc. Fifth National Conf. on Artificial Intelligence*, Philadelphia, PA, August 1986.

Proc. Int'l. Joint Conf. on Neural Networks, San Diego, CA, June 1990.

A Network for Motion Perception[1]

Y. T. Zhou and R. Chellappa

Signal and Image Processing Institute
Department of EE-Systems
University of Southern California
Los Angeles, CA 90089-0272

Abstract

A locally connected artificial neural network based on physiological and anatomical findings in the visual system is presented for motion perception. A set of velocity selective binary neurons is used for each point in the image. Motion perception is carried out by neuron evaluation using a parallel updating scheme. Two algorithms, batch and recursive, based on this network are presented for computing flow field from a sequence of monocular images. The batch algorithm integrates information from all images simultaneously by embedding them into the bias inputs of the network, while the recursive algorithm uses a recursive least squares method to update the bias inputs of the network. Detection rules are also used to find the occluding elements. Based on information on the detected occluding elements, the network automatically locates motion discontinuities. The algorithms need to compute flow field at most twice. Hence, less computations are needed and the recursive algorithm is amenable for real time applications.

1 Introduction

Recently, we have developed an artificial neural network for motion perception based on physiological and anatomical findings in the visual system [1, 2]. The network is discrete, parallel, deterministic and locally connected. A set of velocity selective binary neurons is used for each point in the image. We assume that each neuron receives inputs from itself and other neighboring neurons with similar directional selectivity. Motion perception is carried out by neuron evaluation using a parallel updating scheme.

The network has two important features. First, it can accurately locate motion discontinuities. Usually, a smoothness constraint is used for obtaining a flow field. Using a smoothness constraint may blur surface boundaries and hence motion discontinuities may not be detected. Since motion discontinuities contain rich information about the surface boundaries and the spatial arrangement of the objects, attempts have been made to detect them by using a line process [3]. However, without exactly knowing the occluding elements, the discontinuities detected by the line process may be shifted. We first detect the occluding elements from initial motion measurements and embed them in the bias inputs, then let the network automatically locate the discontinuities. For the purpose of real time implementation, both neurons and lines are updated in a parallel fashion.

Second, our network can use multiple image frames to compute the flow field. Natural images are often degraded by the imaging system. Based on such imperfect observations, it is difficult to compute the flow field accurately, especially near motion depth discontinuities. To improve the accuracy of the solution, multiple frames are used. Two algorithms, batch and recursive, are presented. The batch algorithm simultaneously integrates information from all images by embedding them into the bias inputs of the network, while the recursive algorithm uses a recursive least squares (RLS) method to update the bias inputs of the network. Both these methods need to compute flow field at most twice. Hence, less computations are needed and the recursive algorithm is amenable for real time applications.

[1]This research work is partially supported by the AFOSR Contract No. F-49620-87-C-0007 and the AFOSR Grant No. 86-0196.

2 An Artificial Neural Network

2.1 Physiological Considerations

Microelectrode studies in cats and monkeys indicate that the visual cortex, a few millimeter thick neuronal tissue, is organized in a topographic, laminar and columnar fashion [4, 5]. The image on the retina is first projected to the lateral geniculate bodies and then from there to the visual cortex in a strict topographical manner. The neurons in the visual cortex are arranged in layers and grouped according to several stimulus parameters such as eye dominance, receptive field orientation and receptive field position. The groupings take the form of vertically arranged parallel slabs spanning the full cortical thickness. The optical nerve fibres arriving from the lateral geniculate bodies mostly terminate in layer 4 of visual area 17, yielding a cortical representation of retina. From area 17 the visual signals pass to adjacent area 18 and other higher visual areas such as middle temporal (MT), each with a complete topographic map of the visual field [6, 7, 8].

Figure 1 is an idealized and speculative scheme for the hypercolumns of the visual area 17 [5]. Neurons with similarly orientation- and direction-selectivities are stacked in discrete columns which are perpendicular to the cortical surface. For simplicity, blobs and ocular dominance columns are not included. All the neurons such as simple, complex and hypercomplex within a column have the same receptive field axis orientation. For instance, if an electrode is placed into the cortex in a direction perpendicular to the cortex surface, all the neurons encountered shows the same axis orientation. If the electrode goes in a direction parallel to the cortex surface, there occurs a regular shift in the axis orientation, about $5 - 10^\circ$ for every advance of $25 - 50$ μm. Over a distance of about 1 mm, there is roughly a full rotation (180°). A set of orientation columns representing a full rotation of 180° together with an intersecting pair of ocular dominance columns forms a hypercolumn. Each hypercolumn an elementary unit of the visual cortex is responsible for a certain small area of the visual field and encode a complete feature description of the area by the activity of neurons. Advancing more than 1 mm produces a displacement in the visual field, out of the area where one started and into an entirely new area. The simple neuron is orientation-selective and the complex neuron is direction-selective. The simple neuron responds best to a stationary line which is oriented with the axis of the receptive field. For the complex neurons, not only the orientation of the line but also the stimulus speed and motion direction are important. A oriented line produces strong responses to a complex neuron if it moves at an optimal speed in a direction perpendicular to the receptive field axis orientation within the receptive field. About half of the complex neurons responds only to one direction of movement. If the speed is less or greater than the optimum, the neuron's firing frequency tends to fall off sharply. The optimal speed varies from neuron to neuron. For instance, in cats it varies from about 0.1°/sec up to about 20°/sec [4]. Hence, the complex neurons are direction- and speed-selective, i.e., velocity-selective and area 17 plays a crucial role in determining velocity selection [9, 10]. We assume that the complex neurons within a column can be further grouped according to their speed selectivity. Figure 2 shows a possible 2-D grouping pattern of the complex neurons within a hypercolumn. Each circle represents one or more neurons since several neurons may have the same velocity selectivity. The coordinates of the circle indicate the velocity selectivity of the neurons.

MT is also a "motion area". Neurons in MT are predominantly direction-selective, about 90% show some direction selectivity and 80% are highly selective, and are arranged in columns according to direction selectivity [11, 8, 12]. Area 17 projects to area MT in a very unique way: (1) the projection only happens between the columns with similar directionality; (2) neurons projecting from a given location in area 17 diverge to several periodically spaced locations in MT, and several locations in area 17 converge upon a given location in MT. These properties probably play a very important rule in maintaining axis and direction selectivity and forcing the neighboring receptive fields to have the same directional preference. Figure 3 shows such a projection pattern.

2.2 Computational Considerations

Usually, images are uniformly sampled by the image digitizer and computing flow field is to find the conjugate points in images and compute their displacements. For implementation purposes, we assume that the neurons in a hypercolumn are uniformly distributed over a 2-D Cartesian plane. Then the conjugate point can be found by checking every image pixel within a neighborhood in the successive frame based on the measurement primitives. The maximum search range, i.e., the maximum displacement can be determined by the maximum optimal speed. To improve the accuracy of the solution, the velocity component ranges can be further sampled using bins of size W, where W is a real number.

We assume that each hypercolumn represents a single image pixel or subpixel (if the image is subsampled). If the maximum displacement is D, then about $(2D+1)^2$ mutually exclusive neurons are needed for each pixel and a total number of $N_r \times N_c \times (2D+1)^2$ neurons are required for a $N_r \times N_c$ image. Since two objects cannot occupy the same place at the same time, only one velocity value can be assigned to each pixel. Therefore, in each hypercolumn, only one neuron is in active state. The velocity value can be determined according to its direction selectivity. Figure 4 shows such a network with small frames for the hypercolumns and circles for the neurons. In fact, each small frame contains many neurons. For simplicity, only a few neurons are present in each frame. Each neuron receives a bias input from outside world. The bias input may consist of several different types of measurement primitives, such as the raw image data, filtered image data including their derivatives, edges, lines, corners etc. As the neighboring receptive fields are forced to have the same directional preference, we assume that neurons with similar velocity selectivity in the neighboring hypercolumns tend to affect each other through receiving inputs from each other as shown in Figure 4. This feature implies the smoothness constraint which can be seen more clearly if the network is organized in a multi-layer fashion. Figure 5 shows a multi-layer network which is equivalent to the original one. The network consists of $(2D_k+1) \times (2D_l+1)$ layers. Each layer corresponds to a different velocity and contains $N_r \times N_c$ neurons. Each neuron receives excitatory and inhibitory inputs from itself and other neurons in a neighborhood in the same layer. For each point, only the neuron that has the maximum excitation among all neurons in the other layers is on and the others are off. When the neuron at the point (i,j) in the kth and lth layers is 1, this means that the velocities in k and l directions at the point (i,j) are $k\ W$ and $l\ W$, respectively.

Formally, the multi-layer network can be described as follows. Let $V = \{v_{i,j,k,l}, 1 \le i \le N_r, 1 \le j \le N_c, -D_k \le k \le D_k, -D_l \le l \le D_l\}$ be a binary state set of the neural network with $v_{i,j,k,l}$ denoting the state of the (i,j,k,l)th neuron which is located at point (i,j) in the (k,l)th layer, $T_{i,j,k,l;m,n,k,l}$ the synaptic interconnection strength from neuron (i,j,k,l) to neuron (m,n,k,l) and $I_{i,j,k,l}$ the bias input.

At each step, each neuron (i,j,k,l) synchronously receives inputs from itself and neighboring neurons and a bias input

$$u_{i,j,k,l} = \sum_{(m-i,n-j)\in S_0} T_{i,j,k,l;m,n,k,l} v_{m,n,k,l} + I_{i,j,k,l} \tag{1}$$

where S_0 is an index set for all neighbors in a $\Gamma \times \Gamma$ window centered at point (i,j). The potential of the neuron, $u_{i,j,k,l}$, is then fed back to corresponding neurons after maximum evolution

$$v_{i,j,k,l} = g(u_{i,j,k,l}) \tag{2}$$

where $g(x_{i,j,k,l})$ is a maximum evolution function (it is also called winner-take-all function)

$$g(x_{i,j,k,l}) = \begin{cases} 1 & if\ x_{i,j,k,l} = max(x_{i,j,p,q};\ -D_k \le p \le D_k,\ -D_l \le q \le D_l). \\ 0 & otherwise. \end{cases} \tag{3}$$

The neuron evaluation will be terminated if the network converges, i.e., the energy function of the network defined by

$$E = -\frac{1}{2}\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=-D_k}^{D_k}\sum_{l=-D_l}^{D_l}\Big(\sum_{(m-i,n-j)\in S_0} T_{i,j,k,l;m,n,k,l}\, v_{i,j,k,l}\, v_{m,n,k,l} + I_{i,j,k,l}\, v_{i,j,k,l}\Big). \tag{4}$$

reaches a minimum.

3 Computing Flow Field

A smoothness constraint is used for obtaining a smooth optical flow field and a line process is employed for detecting motion discontinuities. The line process consists of vertical and horizontal lines, L^v and L^h. Each line can be in either one of two states: 1 for acting and 0 for resting. The error function for computing the flow field can be properly expressed as

$$E = \sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=-D_k}^{D_k}\sum_{l=-D_l}^{D_l}\{[A\,(k_{11}(i,j)-k_{21}(i+k,j+l))^2 + A\,(k_{12}(i,j)-k_{22}(i+k,j+l))^2$$
$$+(g_1(i,j)-g_2(i+k,j+l))^2]v_{i,j,k,l} + (B/2)\sum_{s\in S}(v_{i,j,k,l}-v_{(i,j)+s,k,l})^2$$
$$+(C/2)[(v_{i,j,k,l}-v_{i+1,j,k,l})^2(1-L^h_{i,j,k,l}) + (v_{i,j,k,l}-v_{i,j+1,k,l})^2(1-L^v_{i,j,k,l})]\} \quad (5)$$

where $k_{11}(i,j)$ and $k_{12}(i+k,j+l)$ are the principle curvatures of the first image, $k_{21}(i,j)$ and $k_{22}(i+k,j+l)$ are the principle curvatures of the second image, $\{g_1(i,j)\}$ and $\{g_2(i+k,j+l)\}$ are the intensity values of the first and second images, respectively, $S = S_0 - (0,0)$ is an index set excluding $(0,0)$, A, B and C are constants. The principle curvatures can be estimated by using a polynomial fitting technique [1].

The first term in (5) is to seek velocity values such that all points of two images are matched as closely as possible in a least squares sense. The second term weighted by B is the smoothness constraint on the solution and the third term weighted by C is a line process to weaken the smoothness constraint and to detect motion discontinuities. The constant A in the first term determines the relative importance of the intensity values and their principle curvatures to achieve the best results. Note that the error function does not contain a line process penalty term. Suppose we add a penalty term $D(L^h_{i,j,k,l}+L^v_{i,j,k,l})$ to (5) and define the potential of the line, say the vertical line as

$$\varphi_{i,j,k,l} = \frac{C}{2}(v_{i,j,k,l}-v_{i+1,j,k,l})^2 + D. \quad (6)$$

By thresholding the potential $\varphi_{i,j,k,l}$ at zero, the new state of the line takes 1 or 0 accordingly. When $C > 2D$, if two connected neighboring points have different velocity value, then the penalty term can not suppress the line process and a line will be turned on. When $C \leq 2D$, no line will be turned on even if there exists a difference between the two points, which means no penalty is needed. Hence, adding the penalty term or not makes no difference. Basically, the line process does nothing but changing the smoothing weights. For instance, if all lines are on, the weights are all same i.e., $\frac{B}{2}$. If all lines are off, then the weights at the four nearest neighbors of the center point are increased by $\frac{C}{2}$. The line process weakens the smoothness constraints by changing the smoothing weights, resulting in space variant smoothing weights.

By choosing the interconnection strengths and bias inputs as

$$T_{i,j,k,l;m,n,k,l} = -[48B + C(4 - L^h_{i,j,k,l} - L^h_{i,j-1,k,l} - L^v_{i,j,k,l} - L^v_{i-1,j,k,l})]\delta_{i,m}\delta_{j,n}$$
$$+C[(1-L^h_{i,j,k,l})\delta_{i,m}\delta_{j+1,n} + (1-L^h_{i,j-1,k,l})\delta_{i,m}\delta_{j-1,n} + (1-L^v_{i,j,k,l})\delta_{i+1,m}\delta_{j,n}$$
$$+(1-L^v_{i-1,j,k,l})\delta_{i-1,m}\delta_{j,n}] + 2B\sum_{s\in S}\delta_{(i,j),(m,n)+s} \quad (7)$$

and

$$I_{i,j,k,l} = -A[(k_{11}(i,j)-k_{21}(i+k,j+l))^2 + (k_{12}(i,j)-k_{22}(i+k,j+l))^2] - (g_1(i,j)-g_2(i+k,j+l))^2 \quad (8)$$

where $\delta_{a,b}$ is the Dirac delta function, the error function (5) is transformed into the energy function (3) of the neural network. Note that the interconnection strengths consist of constants and line process only. The bias inputs contain all information from the images. This is why we like to call $I_{i,j,k,l}$ bias input instead of threshold, because the neurons receive the information from outside through the bias inputs. Equation (8) also gives us a hint to develop multiple frame algorithms. The size of the smoothing window used in (7) is 5×5.

Computation of flow field is carried out by neuron evaluation. As the first and second terms in (5) do not contain line process, we can then update line process prior to the updating of the neurons. Let $L^{v;new}_{i,j,k,l}$ and $L^{v;old}_{i,j,k,l}$ denote the new and old states of the vertical line $L^v_{i,j,k,l}$, respectively. Let $\varphi_{i,j,k,l}$ be the potential of vertical line $L^v_{i,j,k,l}$ given by

$$\varphi_{i,j,k,l} = \frac{C}{2}(v_{i,j,k,l}-v_{i+1,j,k,l})^2 \quad (9)$$

Then, the new state is determined by

$$L^{v;new}_{i,j,k,l} = \begin{cases} 1 & if \ \varphi_{i,j,k,l} > 0 \\ 0 & otherwise. \end{cases} \tag{10}$$

It is easy to see that whenever the states of neurons $v_{i,j,k,l}$ and $v_{i+1,j,k,l}$ are different, the vertical line $L^v_{i,j,k,l}$ will be active provided that the parameter C is greater than zero. If $C = 0$, then all lines have zero potential and will be inactive. Since the line process weakens the smoothness constraint, the choice of C is closely related to selecting the smoothness parameter B in (5). A similar updating scheme is also used for the horizontal lines.

For each neuron, we use (1) and (2) to synchronously evaluate its state and readjust accordingly. The initial state of the neurons is set as

$$v_{i,j,k,l} = \begin{cases} 1 & if \ I_{i,j,k,l} = max(I_{i,j,p,q}; -D_k \leq p \leq D_k, -D_l \leq q \leq D_l). \\ 0 & otherwise \end{cases} \tag{11}$$

where $I_{i,j,k,l}$ is the bias input. The initial conditions are completely determined by the bias inputs, the information from outside world. If there are two maximal bias inputs at point (i,j), then only the neuron corresponding to the smallest velocity is initially set at 1 and the other one is set at 0. This is consistent with the minimal mapping theory [13]. In the updating scheme, we also use the minimal mapping theory to handle the case of two neurons having the same largest inputs. When the network reaches a stable state, the flow field is determined by the neuron states.

4 Detection of Motion Discontinuities

Motion discontinuities in flow field often result from occluding contours of moving objects. In this case, the moving objects are projected into image plane as adjacent surfaces and their boundaries are undergoing either split or fusion motion [13]. These motion situations give rise to discontinuities along their boundaries. To overcome such difficulties and to locate the discontinuities more accurately, we first detect the occluding elements based on the initial motion measurements. Then all the information about the occluding elements can be embedded into the bias inputs such that the network can automatically take care of motion discontinuities during updating procedure.

Suppose that the surfaces are translating with constant velocities. Let us consider the case in which a surface is moving against a stationary background as shown in Figure 6. Let X_1 denote the occluding element, A_2 and X_2 the corresponding elements of A_1 and X_1, respectively. Let (i,j) be the coordinates of element A_1, d_i and d_j the i and j components of flow at (i,j), respectively. We assume that X_1 and Y_1 are located at $(i+d_i, j+d_j)$ and $(i+2\times d_i, j+2\times d_j)$ respectively. By defining the match errors

$$e_1(i,j) = -I_{i,j,d_i,d_j}, \quad e_2(i,j) = -I_{i+d_i,j+d_j,0,0},$$

$$e_3(i,j) = -I_{i+d_i,j+d_j,d_i,d_j}, \quad e_4(i,j) = -I_{i+2\times d_i,j+2\times d_j,0,0}$$

where $I_{\cdot,\cdot,\cdot,\cdot}$ are bias inputs given in (8), the following relations hold under orthographic or perspective projection for the case when there is no motion along the optical axis,

$$e_1(i,j) \leq e_2(i,j) \quad and \quad e_4(i,j) \leq e_3(i,j). \tag{12}$$

Note that if the above relations do not hold then the element X_1 is not an occluding element. Hence, it is natural to use the relations (12) for detecting the occluding elements.

Detection rule: An occluding element is detected at $(i+d_i, j+d_j)$ if the flow has nonzero values at (i,j),

$$\bar{e}_2(i.j) - \bar{e}_1(i,j) > T \quad \text{and} \quad \bar{e}_4(i.j) - \bar{e}_3(i,j) > T \tag{13}$$

where the theshold T is a nonnegative number and $\bar{e}_k(i,j)$ are the average values of the matching errors within a $\Gamma_T \times \Gamma_T$ window S_T

$$\bar{e}_k(i,j) = \frac{1}{\Gamma_T^2} \sum_{s \in S_T} \bar{e}_k((i,j)+s) \qquad for \quad k = 1,2,3, \ and \ 4.$$

For digitized natural images, T usually takes nonzero value to reduce the effects of quantization error and noise. Using a large value for T can eliminate false occluding elements but may miss the true ones. Since flow at an occluding point has zero values, the *a priori* knowledge about the occluding elements can be embedded in the bias inputs by setting, (for instance at point $(i+d_i, j+d_j)$)

$$I_{i+d_i,j+d_j,0,0} = min(I_{i+d_i,j+d_j,k,l};\ -D_k \leq k \leq D_k, -D_l \leq l \leq D_l). \tag{14}$$

Accordingly, the neural network will prefer zero flow at these points and therefore the line process can precisely locate motion discontinuities.

5 Multiple Frame Approaches

When image quality is poor, measurement primitives estimated from these images are not accurate and reliable. For instance, if the images are blurred by motion, then the local features are smeared and much of the information is lost. Especially, the object boundaries become wide and the derivatives of the intensity function become small at the boundaries. Based on these low quality measurements, motion discontinuities can not be correctly located and hence flow field can not be accurately computed. To improve the accuracy, one way is to improve the image quality by using some image restoration techniques to remove degradations. However, without *a priori* knowledge of the degradations, such as blur function, an image can not be restored perfectly. When the blur is ill conditioned, it is still difficult to restore the image even if the blur function is given. An alternative is to compute flow field over a long time interval, i.e. using multiple frames. In this section, two algorithms, batch and recursive, using more than two frames of images are presented.

5.1 Batch Algorithm

Assume that the objects in the scene are moving with a constant velocity translational motion. It is interesting to note that in the two frame case the bias inputs (8) contain nothing but the measurement primitives, the intensity values and principle curvatures, which are estimated from images. The bias inputs of the network are completely determined by the observations, i.e. the images, while the interconnection strengths (7) do not contain any observations. All these facts suggest that any information from outside world can be included in the bias inputs. The network learns all information directly from the inputs. Hence, it is natural to extend the two frame approach to multiple frames by adding more observations to the bias inputs. Assuming M frames of images are available, the bias inputs are given by

$$\begin{aligned} I_{i,j,k,l} \;=\; & -\sum_{r=1}^{M-1} \{A[(k_{r1}(i+rk-k, j+rl-l) - k_{(r+1)1}(i+rk, j+rl))^2 + (k_{r2}(i+rk-k, j+rl-l) \\ & -k_{(r+1)2}(i+rk, j+rl))^2] + (g_r(i+rk-k, j+rl-l) - g_{r+1}(i+rk, j+rl))^2\}. \end{aligned} \tag{15}$$

Accordingly, the initial state of the neurons (11) is set by using these new bias inputs. The two frame algorithm presented in section 2.2 can be used without any modifications for the multiple frame case.

5.2 Recursive Algorithm

If all the images are not available at the same time or if one wants to compute the flow field in real time, a recursive algorithm can be used. The recursive algorithm uses an RLS algorithm to update the bias inputs. First, the initial condition for the bias inputs is set to zero, i.e. $I_{i,j,k,l}(0) = 0$. This is reasonable, because there is no information available at the beginning. Then, whenever a new frame becomes available, the bias inputs can be updated by

$$I_{i,j,k,l}(r) = I_{i,j,k,l}(r-1) + \frac{1}{r}(\tilde{I}_{i,j,k,l}(r) - I_{i,j,k,l}(r-1)) \qquad for \quad 2 \leq r \leq M \tag{16}$$

where $\tilde{I}_{i,j,k,l}(r)$ is a new observation given by

$$\begin{aligned}\tilde{I}_{i,j,k,l}(r) &= -A[(k_{r1}(i+rk-k,j+rl-l)-k_{(r+1)1}(i+rk,j+rl))^2+(k_{r2}(i+rk-k,j+rl-l)\\ &\quad -k_{(r+1)2}(i+rk,j+rl))^2]+(g_r(i+rk-k,j+rl-l)-g_{r+1}(i+rk,j+rl))^2. \qquad (17)\end{aligned}$$

In fact, this RLS algorithm is equivalent to the batch algorithm. If the intermediate results are not required, flow field can be computed after all images are received. As one can see, the RLS algorithm is parallel in nature and very few computations are required at each step. Hence this algorithm is extremely fast and can be implemented in real time.

Since the number of occluding elements dramatically increases as more image frames are involved, special attention has to be paid to this problem. With minor modifications, the detection criterion used for the two frames can be extended to multiple frames [1].

The multiple frame based algorithms can be summarized as

1. Compute flow field from the first two frames.
2. Update bias inputs.
3. Detect occluding elements and reset bias inputs accordingly. For the recursive algorithm, go back to step 2 if the incoming frame is not the last one; otherwise, go to step 4.
4. Compute flow field with updated bias inputs.

6 Results

Figure 7 shows the first and fourth frames of a sequence of images, a pick-up truck moving from right to left against a stationary background. The images are of size 480 x 480. As the rear part of the truck is missing in the first frame, we reversed the order of image sequence so that there is a complete truck image in the first frame. Accordingly, the direction of the computed flow field should be reversed. For the two frame approach, we used the fourth frame as the first frame and the third frame as the second frame. For reducing computations, the image size was reduced to 120 x 120 by subsampling.

For each point, we use two memories in the range $-D_k$ to D_k and $-D_l$ to D_l to represent velocities in i and j directions, respectively, instead of using $(2D_k+1)$ and $(2D_l+1)$ neurons. Due to local connectivity, the potential of the neuron is computed only within a small window. By setting $A=2$, $B=250$, $C=50$, $D_k=7$, and $D_l=1$, the flow field was obtained after 36 iterations. A 48 x 113 sample of the computed flow field corresponding to the part framed by black lines in Figure 7(b) is given in Figure 8. Since the shutter speed was low, the truck was heavily blurred by the motion. The motion blur smeared the edges and erased local features, especially the features on the wheel. Hence, it is difficult to detect the rotation of the wheels. Also note that although most of the boundary locations are correct, the boundaries due to the fusion motion such as the rear part of the truck and the driver's cab are shifted by the line process.

The occluding pixels were detected at $T=100$ based on the initially computed flow field of Figure 8. By embedding the information about the occluding pixels into the bias inputs, using the initially computed optical flow as the initial conditions and choosing $A=2$, $B=188$, $C=200$, $D_k=7$ and $D_l=1$ the final result shown in Figure 9 was obtained after 13 iterations. The accuracy of boundary location is significantly improved.

For the multiple frame approaches, we used four image frames. Theoretically there is no limit to the number of frames that can be used in the batch approach. For the same reason mentioned before, the fourth frame was taken as the first frame, the third frame as the second frame, etc. Since the batch and recursive algorithms are similar in spirit, a set of identical parameters was used for both algorithms in this experiment. and the same results were obtained. The occluding pixels were also detected at $T=100$ from four frames. Figure 10 shows the flow field computed from four frames using the occluding pixel information. The parameters used were $A=4$, $B=850$, $C=80$, $D_k=7$ and $D_l=1$, and 12 iterations were required. As expected, the output is much cleaner and the boundaries are more accurate than that of the two frame based approach. The number of iterations is also reduced.

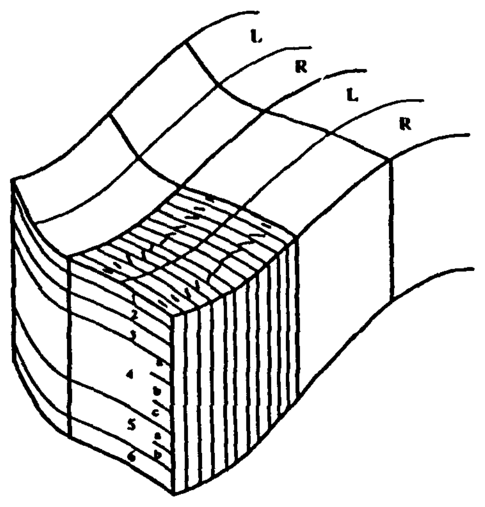


Figure 1: An idealized and speculative scheme for the hypercolumns of visual area 17. The blocks denoted by thick lines represent hypercolumns containing complete sets of orientation columns. The thin lines separate individual orientation columns. L and R denote the areas corresponding to left and right eyes.

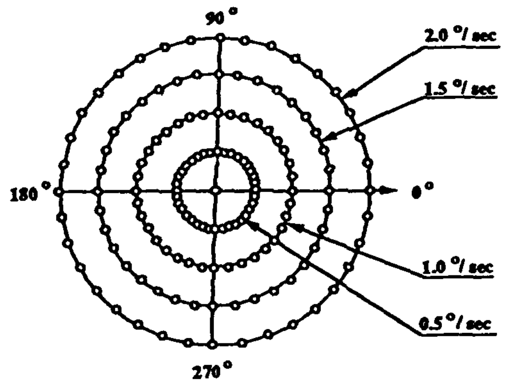


Figure 2: A 2-D grouping pattern for the neurons within a hypercolumn. Neurons are arranged according to their direction and speed selectivity.

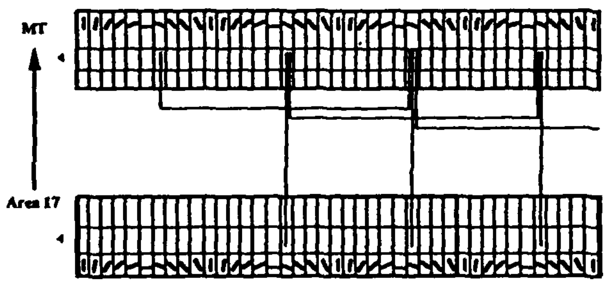


Figure 3: Projection pattern from area 17 to MT. For simplicity, only the cross-section of the four different orientation selective hypercolumns are shown for each area.

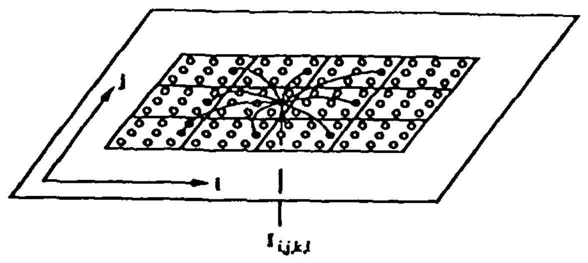


Figure 4: An artificial neuron network. Small frame denotes the hypercolumn. The neurons in a hypercolumn are uniformly distributed on a plane.

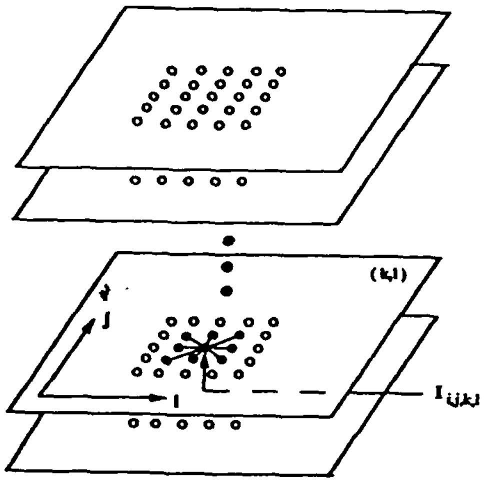


Figure 5: A multi-layer network which is equivalent to the original one. The neurons are arranged in layers according to their velocity selectivity. i and j denote the image coordinates. k and l denote the velocity coordinates.

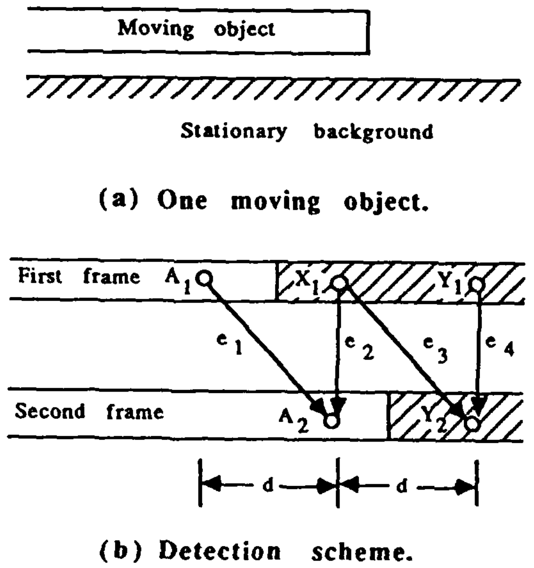


Figure 6: Detection of the occluding element.

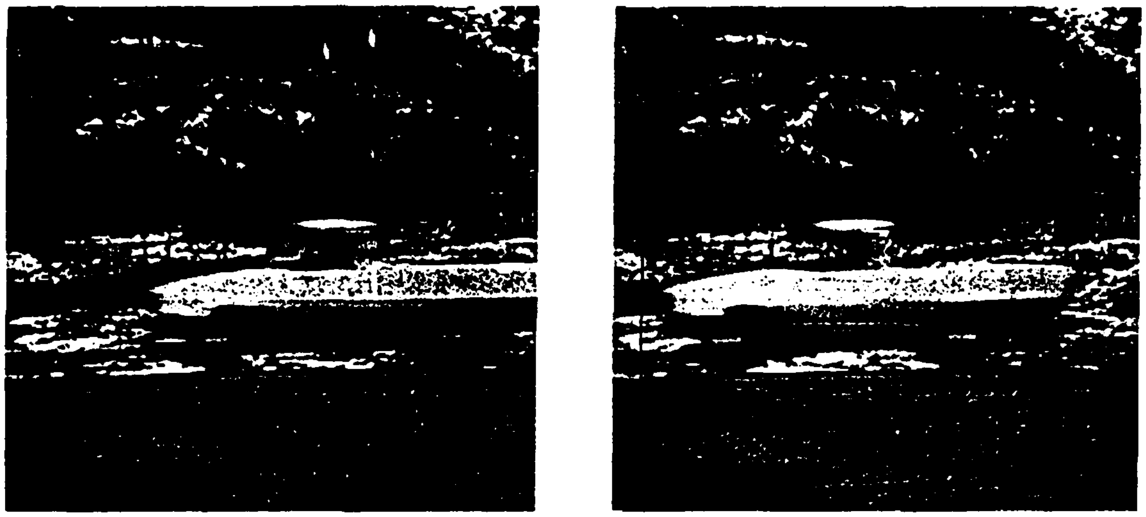

(a) (b)

Figure 7: A sequence of pick-up truck images. (a) The first frame. (b) The fourth frame.

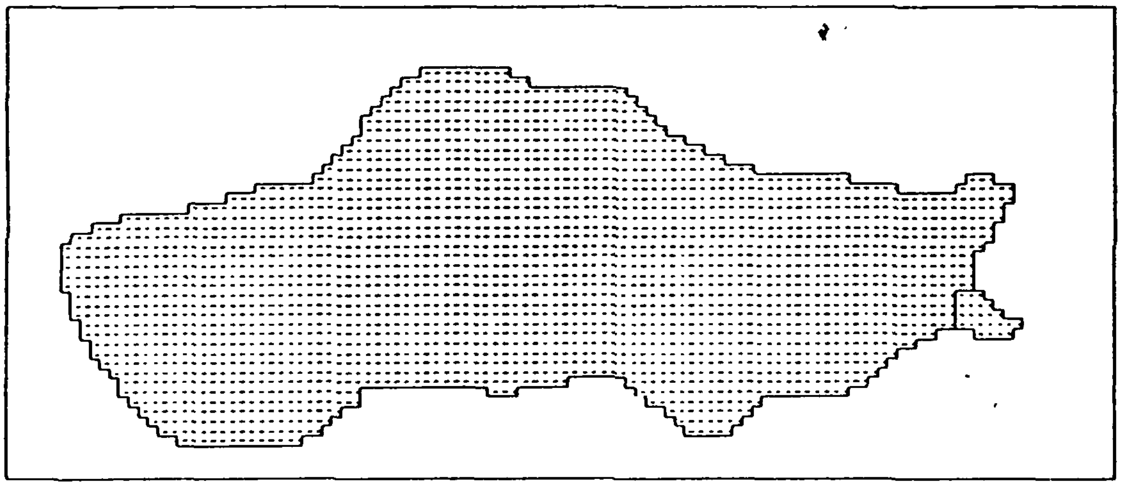


Figure 8: Flow field computed from two frames.

References

[1] Y. T. Zhou. *"Artificial Neural Network Algorithms for Some Computer Vision Problems"*. PhD thesis, University of Southern California, Los Angeles, CA, November 1988.

[2] Y. T. Zhou and R. Chellappa. "neural network algorithms for motion stereo". In *Proc. Intl. Joint Conf. on Neural Networks*, volume vol. 2, pages 251–258, Washington D.C., June 1989.

[3] C. Koch. "analog neuronal networks for real-time vision systems". In *Proc. Workshop on Neural Network Devices and Applications*, Los Angeles, CA, February 1987.

[4] D. H. Hubel and T. N. Wiesel. "receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat". *J. Neurophysiol.*, vol. 128:229–289, 1965.

[5] D. H. Hubel and T. N. Wiesel. "functional architecture of macaque monkey visual cortex". *Proc. R. Soc. Lond. Ser. B*, vol. 198:1–59, 1977.

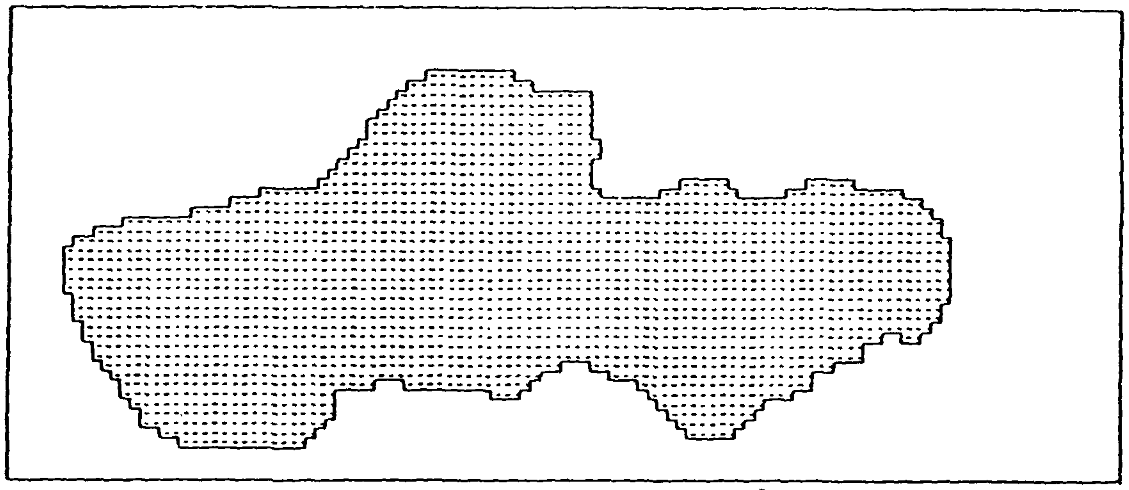


Figure 9: Flow field computed from two frames using the occluding element information.

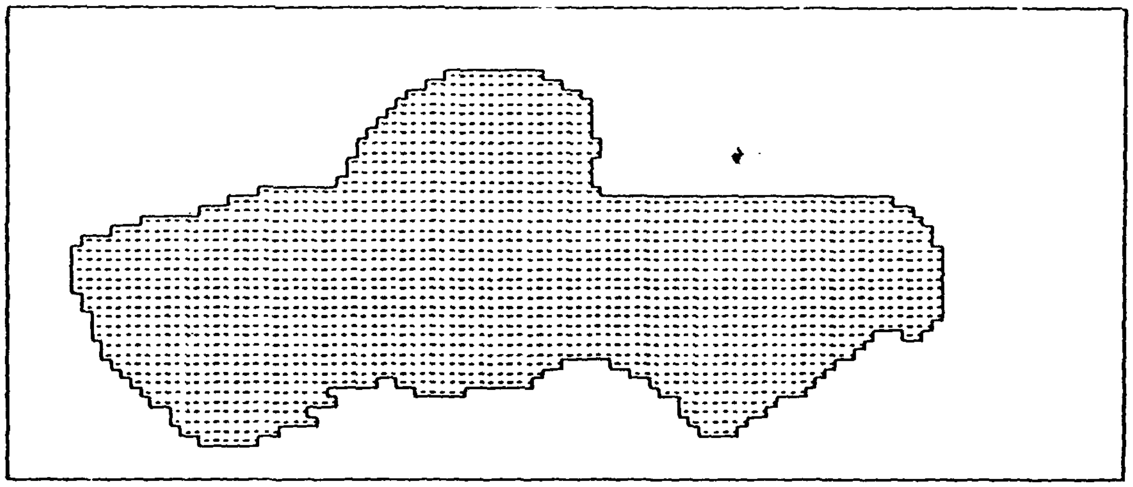


Figure 10: Flow field computed from four frames using the occluding element information.

[6] J. Tiggs, M. Tiggs, S Anschel, N.A. Cross, W. D. Ledbetter, and R. L. McBride. "areal and laminar distribution of neurons interconnecting the central visual cortical areas 17, 18 19 and mt in squirrel monkey (saimiri)". *J. Comp. Neurol.*, vol. 202:539–560, 1981.

[7] J. F. Baker, S. E. Petersen, W.T. Newsome, and J. M. Allman. "visual response properties of neurons in four extrastriate visual areas of the owl monkey (aotus trivirgatus): A quantitative comparison of medial, dorsomedial, and middle temporal areas". *J. Neurophysiol.*, vol. 45:397–416, 1981.

[8] S. Lin C, R. E. Weller, and J. H. Kaas. "cortical connections of striate cortex in the owl monkey". *J. Comp. Neurol.*, vol. 211:165–176, 1982.

[9] A. C. Rosenquist and L. A. Palmer. "visual receptive field properties of cells of the superior colliculus after cortical lesions in the cat". *Exp. Neurol.*, vol. 33:629–652, 1971.

[10] K. Ogasawara, J. G. McHaffie, and B. E. Stein. "two visual corticotectal systems in cat". *J. Neurophysiol.*, vol. 52:1226–1245, 1984.

[11] V. M. Montero. "patterns of connections from striate cortex to cortical visual area in superior temporal sulcus of macaque and middle temporal gyrus of owl monkey". *J. Comp. Neurol.*, vol. 189:45–59, 1980.

[12] T. D. Albright, R. Desimone, and C. G. Gross. "columnar organization of directionally selective cells in visual area mt of the macaque". *J. Neurophysiol.*, vol. 51:16–31, 1984.

[13] S. Ullman. *The Interpretation of Visual Motion.* M.I.T. Press, Cambridge, MA, 1979.

Reprint from PROCEEDINGS FROM ICASSP — INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, New York, N.Y., April 11-14, 1988

STEREO MATCHING USING A NEURAL NETWORK[1]

Y. T. Zhou and R. Chellappa

Signal and Image Processing Institute
Department of EE-Systems
University of Southern California

Abstract

A method for matching stereo images using a neural network is presented. Usually, the measurement primitives used for stereo matching are the intensity values, edges and linear features. Conventional methods based on such primitives suffer from amplitude bias, edge sparsity and noise distortion. We first fit a polynomial to find a smooth continuous intensity function in a window and estimate the first order intensity derivatives. A neural network is then employed to implement the matching procedure under the epipolar, photometric and smoothness constraints based on the estimated first order derivatives. Owing to the dense intensity derivatives a dense array of disparities are generated with only a few iterations. This method does not require surface interpolation. Computer simulations to demonstrate the efficacy of our method are presented.

1 Introduction

Stereo matching is a primary means for recovering 3–D depth from two images taken from different viewpoints. The two central problems in stereo matching are to match the corresponding points and to obtain a depth map or disparity values between these points. In this paper we present a method for computing the disparities between the corresponding points in two images recorded simultaneously from a pair of laterally displaced cameras based on the first order intensity derivatives. An implementation using a neural network is also given.

Basically, there exist two types of stereo matching methods: region based and feature based methods according to the nature of the measured primitives. The region based methods use the intensity values as the measurement primitives. A correlation technique or some simple modification is applied to certain local region around the pixel to evaluate the quality of matching. The region based methods usually suffer from the problems due to lack of local structures in homogeneous regions, amplitude bias between the images and noise distortion. Recently, Barnard [1] applied a stochastic optimization approach for the stereo matching problem to overcome the difficulties due to homogeneous regions and noise distortion. Although this approch is different from the conventional region based methods, it still uses intensity values as the primitives with the aid of a smoothness constraint. Barnard's approach has several advantages: simple, suitable for parallel processing and a dense disparity map output. However, too many iteations, a common problem with the simulated annealing algorithm, makes it unattractive. It also suffers from the problem of amplitude bias between the two images.

The feature based methods use intensity edges or linear features (for example, see Grimson [2] and Medioni [3]) or intensity peaks which correspond to discontinuities in the first order derivatives of intensity [4]. The intensity edges are obtained using edge detectors such as the Marr–Hildreth edge detector [5] or the Nevatia–Babu line finder [6]. Since amplitude bias and small amount noise do not affect edge detection, feature based methods can handle natural images more efficiently. Owing to fewer measurement primitives to deal with, usually feature based methods are faster than the region based methods. However, a surface interpolation step has to be included. In order to obtain smooth surface interpolations, several types of smoothing constraint techniques have been introduced [2]. The common problem in feature based methods is that if features are sparse, then surface interpolation step is usually difficult.

Julesz's example of random dot stereograms shows that stereo matching occurs very early in the visual process [7]. The question is what kind of measurement primitives human stereo process does use. As we know that the amplitude bias can be eliminated by differential operation, the intensity derivatives are dense, and human visual system is sensitive to the intensity changes, the first order intensity derivatives (simplest derivatives) may be considered as appropriate measurement primitives for the stereo matching problem. Noise distortion, which the first order derivatives are very sensitive to, can be reduced by some smoothing techniques such as polynomial fitting technique. The first order intensity derivatives can be obtained by directly taking the derivative about the resulting continuous intensity function.

Recently, many researchers have been using neural network for stereo matching based on either intensity values or edges [8,9,10]. Early work on extracting the depth information from the random dot stereogram using neural network can be found in [11]. In this paper, we use a neural network with maximum evolution function to solve the stereo matching problem based on the first order intensity derivatives under the epipolar, photometric and smoothness constraints. We illustrate the usefulness of this approach by using the random dot stereograms. Due to lack of space, natural image examples are not given in this paper.

2 Estimation of the First Order Intensity Derivatives

Natural digital images usually are corrupted by certain amount of niose due to electronic imaging sensor, film granularity and quantization error. The derivatives obtained using a difference operator applied to digital images are not reliable. Since digital image comes about by sampling an analog image on an equally spaced lattice, a proper way to recover a smooth and continuous image surface is by a polynomial fitting technique. We first assume that, a point at the right image corresponding to a specified point in the left image lies somewhere on the corre-

[1]This research work is partially supported by the AFOSR Contract No. F-49620-87-C-0007 and the AFOSR Grant No. 86-0196.

sponding epipolar line which is parallel to the row coordinate, i.e. in a horizontal direction, and second, in each neighborhood of image the underlying intensity function can be sufficiently approximated by a 4th order polynomial. The first assumption is also known as the epipolar constraint. With the help of this constraint, the first order derivatives we need for matching are computed only for the horizontal direction. Under the second assumption, the intensity funtion in a window, centered at the point (i,j), of size $2r+1$ is fitted by a polynomial of the form

$$g(i,j+x) = a_1 + a_2x + a_3x^2 + a_4x^3 + a_5x^4 \tag{1}$$

where x is lies in the range $-r$ to $+r$ and $\{a_i\}$ are coeffecients. The first order intensity derivative at point (i,j) can easily be obtained by taking the derivative about $g(i,j+x)$ with respect to x and then setting $x=0$

$$\frac{dg(i,j)}{dj} \triangleq \frac{dg(i,j+x)}{dx}\Big|_{x=0} = a_2 \tag{2}$$

In order to estimate each coeffecient independently, an orthogonal polynomial basis set is used. Several existing orthogonal polynomial basis sets can be found in [12,13]. We use the discrete Chebyshev polynomial basis set, also used by Haralick for edge detection and topographic classification [14,15]. The important property of using polynomials is that low order fits over a large window can reduce the effects of noise and give a smooth function.

Let a set of discrete Chebyshev polynomials be defined over an index set $R = \{-r, -r+1, ..., r-1, r\}$, i.e. over a window of size $2r+1$, as

$$\begin{aligned} Ch_0(x) &= 1 \\ Ch_1(x) &= x \\ Ch_2(x) &= x^2 - \tfrac{q_2}{q_0} \\ Ch_3(x) &= x^3 - \tfrac{q_4}{q_2}x \\ Ch_4(x) &= x^4 + \tfrac{q_2q_4-q_6}{q_0q_4-q_2}x^2 + \tfrac{q_2q_6-q_4^2}{q_0q_4-q_2} \end{aligned} \tag{3}$$

where

$$q_n = \sum_{k\in R} k^n.$$

With the window centered at point (i,j), the intensity function $g(i,j+x)$ for each $x \in R$ can be obtained as

$$\hat{g}(i,j+x) = \sum_{m=0}^{4} d_m\, Ch_m(x) \tag{4}$$

where $\hat{g}(i,j+x)$ denotes the approximated continuous intensity funtion. By minimizing the least square error in estimation and taking advantage of the orthogonality of the polynomial set, the coefficients $\{d_m\}$ are obtained as

$$d_m = \frac{\sum_{x\in R} Ch_m(x)\, g(i,j+x)}{\sum_{u\in R} Ch_m^2(u)} \tag{5}$$

where $\{g(i,j+x)\}$ are the observed data values.

Extending (4) and comparing with terms in (1), the first order intensity derivative coefficient a_2, is given by

$$\begin{aligned} a_2 &= d_1 - \frac{q_4}{q_2}d_3 \\ &= \sum_{x\in R} M(x)\, g(i,j+x) \end{aligned} \tag{6}$$

where $M(x)$ is determined by

$$M(x) = \frac{Ch_1(x)}{\sum_{u\in R} Ch_1^2(u)} - \frac{q_4\, Ch_3(x)}{q_2 \sum_{u\in R} Ch_3^2(u)} \tag{7}$$

From (6) one can see that $M(x)$ is a filter for detecting intensity changes.

3 A Neural Network for Matching

We use a neural network containing binary neurons for representing the disparity values between the two images. The model consists of $N_r \times N_c \times D$ mutually interconnected neurons, where D is the maximum disparity, N_r and N_c are the image row and column sizes, respectively. Let $V = \{v_{i,j,k}, 1 \le i \le N_r, 1 \le j \le N_c, 0 \le k \le D\}$ be a binary state set of the neural network with $v_{i,j,k}$ (1 for firing and 0 for resting) denoting the state of the (i,j,k)th neuron. Especially, when the neuron $v_{i,j,k}$ is 1, this means that the disparity value is k at the point (i,j). Every point is represented by $D+1$ mutually exclusive neurons, i.e. only one neuron is firing and others are resting, due to the uniqueness constraint of the matching problem. Let $T_{i,j,k;l,m,n}$ denote the strength (possibly negative) of the interconnection between neuron (i,j,k) and neuron (l,m,n). We require symmetry

$$T_{i,j,k;l,m,n} = T_{l,m,n;i,j,k}$$

$$for \quad 1 \le i,l \le N_r \quad 1 \le j,m \le N_c \quad and \quad 0 \le k,n \le D$$

We also insist that the neurons have self-feedback, i.e. $T_{i,j,k;i,j,k} \neq 0$. In this model, each neuron (i,j,k) randomly and asynchronously receives inputs $\sum_{l,m,n} T_{i,j,k;l,m,n} v_{l,m,n}$ from all neurons and a bias input $I_{i,j,k}$

$$u_{i,j,k} = \sum_{l=1}^{N_r} \sum_{m=1}^{N_c} \sum_{n=0}^{D} T_{i,j,k;l,m,n} v_{l,m,n} + I_{i,j,k} \tag{8}$$

Each $u_{i,j,k}$ is fed back to corresponding neurons after maximum evolution

$$v_{i,j,k} = g(u_{i,j,k}) \tag{9}$$

where $g(x_{i,j,k})$ is a nonlinear maximum evolution function whose form is taken as

$$g(x_{i,j,k}) = \begin{cases} 1 & if \ \ x_{i,j,k} = max(x_{i,j,l}; l = 0,1,...,D). \\ 0 & otherwise. \end{cases} \tag{10}$$

In this model, the state of each neuron is updated by using the latest information about other neurons. The uniqueness of matching problem is ensured by a batch updating scheme–$D+1$ neurons are updated at each step simultaneously.

3.1 Estimation of Model Parameters

The neural model parameters, the interconnection strengths and the bias inputs, can be determined in terms of the energy function of the neural network. As defined in [16], the energy function of the neural network can be written as

$$\begin{aligned} E = &-\frac{1}{2}\sum_{i=1}^{N_r}\sum_{l=1}^{N_r}\sum_{j=1}^{N_c}\sum_{m=1}^{N_c}\sum_{k=0}^{D}\sum_{n=0}^{D} T_{i,j,k;l,m,n}\, v_{i,j,k}\, v_{l,m,n} \\ &-\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=0}^{D} I_{i,j,k}\, v_{i,j,k} \end{aligned} \tag{11}$$

In order to use the spontaneous energy-minimization process of the neural network, we reformulate the stereo matching problem under the epipolar assumption as one of minimizing an error function with constraints defined as

$$E = \sum_{i=1}^{N_r}\sum_{j}^{N_c}\sum_{k=0}^{D} (g'_L(i,j) - g'_R(i,j\oplus k))^2\, v_{i,j,k}$$

$$+\frac{\lambda}{2}\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=0}^{D}\sum_{s\in S}(v_{i,j,k}-v_{i,j\oplus s,k})^2 \quad (12)$$

where $\{g_L'(\cdot)\}$ and $\{g_R'(\cdot)\}$ are the first order intensity derivatives of the left and right images, respectively, S is an index set for all neighbors in a 5×5 window centered at point (i,j), λ is a constant and the symbol $\oplus$ denotes that

$$f_{a\oplus b}=\begin{cases} f_{a+b} & if\ 0\le a+b\le N_c, N_r \\ 0 & otherwise\end{cases}$$

The first term called the photometric constriant in (12) is to seek disparity values such that all regions of two images are matched as close as possible in a least squares sense. Meanwhile, the second term is the smoothness constraint on the solution. The constant λ determines the relative importance of the two terms to achieve the best results.

By comparing the terms in the expansion of (12) with the corresponding terms in (11), we can determine the interconnection strengths and bias inputs as

$$T_{i,j,k;l,m,n}=-48\lambda\delta_{i,l}\delta_{j,m}\delta_{k,n}+2\lambda\sum_{s\in S}\delta_{i,l}\delta_{j,m\oplus s}\delta_{k,n} \quad (13)$$

and

$$I_{i,j,k}=-(g_L'(i,j)-g_R'(i,j\oplus k))^2 \quad (14)$$

where $\delta_{a,b}$ is the Dirac delta function.

3.2 Stereo Matching

Stereo matching is carried out by neuron evaluation. Once the parameters $T_{i,j,k;l,m,n}$ and $I_{i,j,k}$ are obtained using (13) and (14), each neuron can randomly and asynchronously evaluate its state and readjust accordingly using (8) and (9).

The initial state of the neurons were set as:

$$v_{i,j,k}=\begin{cases} 1 & if\ I_{i,j,k}=max(I_{i,j,l};l=0,1,...,D). \\ 0 & otherwise\end{cases} \quad (15)$$

where $I_{i,j,k}$ is the bias input.

However, this neural network has self-feedback, i.e. $T_{i,j,k;i,j,k}\neq 0$, as a result of a transition the energy function E does not decrease monotonically. This is explained as follows. Since we are using a batch updating scheme, $(D+1)$ nerons $\{v_{i,j,k};k=0,...,D\}$ corresponding to the image point (i,j) are simultaneously updated at each step. However, at most two of the $(D+1)$ neurons change their state at each step. Define the state changes $\Delta v_{i,j,k}$ and $\Delta v_{i,j,k'}$ of neurons (i,j,k) and (i,j,k') and energy change ΔE as

$$\Delta v_{i,j,k}=v_{i,j,k}^{new}-v_{i,j,k}^{old}$$

$$\Delta v_{i,j,k'}=v_{i,j,k'}^{new}-v_{i,j,k'}^{old}$$

and

$$\Delta E=E^{new}-E^{old}$$

Consider the energy function

$$\begin{aligned} E &= -\frac{1}{2}\sum_{i=1}^{N_r}\sum_{l=1}^{N_r}\sum_{j=1}^{N_c}\sum_{m=1}^{N_c}\sum_{k=0}^{D}\sum_{n=0}^{D}T_{i,j,k;l,m,n}\,v_{i,j,k}\,v_{l,m,n} \\ &\quad -\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\sum_{k=0}^{D}I_{i,j,k}\,v_{i,j,k}, \quad (16)\end{aligned}$$

the change ΔE due to a changes $\Delta v_{i,j,k}$ and $\Delta v_{i,j,k'}$ given by

$$\begin{aligned}\Delta E &= -(\sum_{l=1}^{N_r}\sum_{m=1}^{N_c}\sum_{n=0}^{D}T_{i,j,k;l,m,n}\,v_{l,m,n}+I_{i,j,k}\,)\,\Delta v_{i,j,k} \\ &\quad -(\sum_{l=1}^{N_r}\sum_{m=1}^{N_c}\sum_{n=0}^{D}T_{i,j,k';l,m,n}\,v_{l,m,n}+I_{i,j,k'}\,)\,\Delta v_{i,j,k'} \\ &\quad -\frac{1}{2}T_{i,j,k;i,j,k}\,(\Delta v_{i,j,k})^2-\frac{1}{2}T_{i,j,k';i,j,k'}\,(\Delta v_{i,j,k'})^2 \\ &\quad -T_{i,j,k;i,j,k'}(\Delta v_{i,j,k}v_{i,j,k'}^{new}+\Delta v_{i,j,k'}v_{i,j,k}^{new}) \quad (17)\end{aligned}$$

is not always negative. A simple proof for this may be found in [17]. Therefore, the convergence of the network is not guaranteed [18].

To ensure convergence of the network, we have designed a deterministic decision rule. The rule is to take a new state $v_{i,j,k}^{new}$ and $v_{i,j,k'}$ of neurons (i,j,k) and (i,j,k') if the energy change ΔE due to state changes $\Delta v_{i,j,k}$ and $\Delta v_{i,j,k'}$ is less than zero. If ΔE due to state change is >0, no state change is affected. A stochastic decision rule can also be used to obtain a globally optimal solution [19].

The stereo matching algorithm can then be summarized as

1. Set the initial state of the neurons.
2. Update the state of all neurons asynchronously and randomly according to the decision rule.
3. Check the energy function; if energy does not change anymore, stop; otherwise, go back to step 2.

4 Experimental Results

A variety of images including random dot stereograms and natural stereo image pairs were tested using our algorithm. The random dot stereograms were created by the pseudo random number generating method described in [20]. Each dot consists of only one element. All the following random dot stereograms are in the form of three level "wedding cake". The background plane has zero disparity and each successive layer plane has additional two elements of disparity. In order to implement this algorithm more efficiently on a conventional computer, we make the following simplifications. Since only one of $D+1$ neurons is firing at each point, we used one neuron lying in the range 0 to D to represent the disparity value instead of $D+1$ neurons. From (13) one can see that the interconnections between the neurons are local (a 5×5 neighborhood) and have the same structure for all neurons. Therefore, we used a 5×5 window for computing $U_{i,j,k}$ and energy function E instead of a $N_rN_c(D+1)\times N_rN_c(D+1)$ interconnection strength matrix. The simplified algorithm greatly reduces the space complexity by increasing the program complexity little. Therefore, it is very fast and efficient.

Figure 1 shows a 10% random dot stereogramIntensity values of the white and black elements are 255 and 0, respectively. Figure 1(c) is the resulting disparity map after 14 iterations. The disparity values are encoded as intensity values with the brightest value denoting the maximum disparity value. We used $\lambda=20$, $D=6$ and $r=2$ (i.e. window size was 5). Note that the disparity map is dense.

A similar test was run on the decorrelated stereogram [7]. The original stereogram is 50% density random dots. In the left image, 20% of the dots were decorrelated at random. By setting $\lambda=2800$, $D=6$ and $r=2$, a dense disparity map in Figure 2(c) was obtained after 12 iterations.

Owing to page limitations, natural image examples are not given in this paper.

References

[1] S. T. Barnard, "A Stochastic Approach to Stereo Vision", In *Proc. Fifth National Conf. on Artificial Intelligence*, Philadelphia, PA, August 1986.

[2] W. E. L. Grimson, *From Images to Surfaces*, The MIT press, Cambridge, MA, 1981.

[3] G. Medioni and R. Nevatia, "Segment-Based Stereo Matching", *Computer Vision, Graph. and Image Processing*, vol. 31, pp. 2-18, 1985.

[4] J. E. W. Mayhew and J. P. Frisby, "Psychophysical and Computational Studies towards a Theory of Human Stereopsis", *Artificial Intelligence*, vol. 17, pp. 349-385, August. 1981.

[5] D. Marr and E. C. Hildreth, "Theory of Edge Detection", *Proc. Royal Society of London*, vol. B-207, pp. 187-217, February 1980.

[6] R. Nevatia and K. R. Babu, "Linear Feature Extraction and Description", *Computer Graph. and Image Processing*, vol. 13, pp. 257-269, 1980.

[7] B. Julesz, *Foundations of Cyclopean Perception*, The University of Chicago Press, Chicago, IL, 1971.

[8] N. M. Grzywacz and A. L.Yuille, "Motion Correspondence and Analog Networks", In *Proc. Conf. on Neural Networks for Computing*, pp. 200-205, American Institute of Physics, Snowbird, UT, 1986.

[9] G. Z. Sun, H. H. Chen, and Y. C. Lee, "Learning Stereopsis with Neural Networks", In *Proc. IEEE First Annual Intl. Conf. on Neural Networks*, San Diego, CA, June 1987.

[10] A. F. Gmitro and G. R. Gindi, "Optical Neurocomputer for Implementation of the Marr-Poggio Stereo Algorithm", In *Proc. IEEE First Annual Intl. Conf. on Neural Networks*, San Diego, CA, June 1987.

[11] P.Dev, "Perception of Depth Surfaces in Random-dot Stereogram: a Neural Model", *Int. J. Man-Machine Studies*, 7, pp. 511-528, 1975.

[12] G. G. Lorentz, *Approximation of functions*, Holt, Rinehart and Winston, New York, 1966.

[13] P. Beckmann, *Orthogonal Polynomials for Engineers and Physicists*, Golem, Boulder, CO, 1973.

[14] R. M. Haralick, "Digital Step Edges from Zero Crossings of Second Directional Derivatives", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-6, pp. 58-68, January 1984.

[15] T. J. Laffey, R. M. Haralick, and L. T. Watson, "Topographic Classification of Digital Image Intensity Surfaces", In *The proc. IEEE Workshop on Computer Vision: Theory and Control*, Rindge, New Hampshire, August 1982.

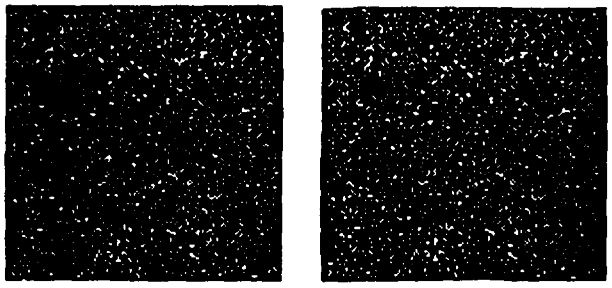
[16] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, vol. 52, pp. 114-152, 1985.

[17] Y. T. Zhou and R. Chellappa, *Stereo Matching Using a Neural Network*, Technical Report USC SIPI Report, Signal and Image Processing Institute, Univ. of Southern California, December 1987.

[18] J. P. LaSalle, *The Stability and Control of Discrete Processes*, Springer-Verlag, New York, New York, 1986.

[19] Y. T. Zhou, R. Chellappa, and B. K. Jenkins, "A Novel Approach to Image Restoration Based on a Neural Network", In *Proc. IEEE First Annual Intl. Conf. on Neural Networks*, San Diego, CA, June 1987.

[20] B. Julesz, "Binocular Depth Perception of Computer-Generated Patterns", *Bell System Technical J.*, vol. 39, pp. 1125-1162, Sept. 1960.


(a) Left image. (b) Right image.

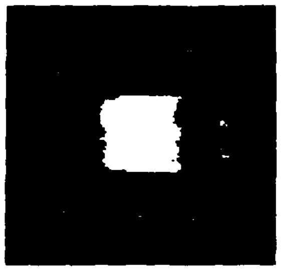
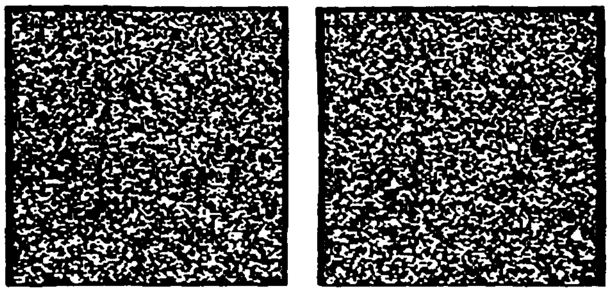

(c) Disparity map represented by intensity image.

Figure 1: A 10% density random dot stereogram.


(a) Left image. (b) Right image.

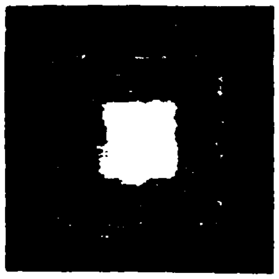

(c) Disparity map represented by intensity image.

Figure 2: A 50% density random dot stereogram. In the left image, 20% of the dots were decorrelated at random.

Computational Issues in Digital Optical Computing

B. Keith Jenkins
Signal and Image Processing Institute MC-0272
University of Southern California, Los Angeles, California 90089-0272

and

C. Lee Giles
Air Force Office of Scientific Research/NE
Bolling AFB, D.C. 20032-6448

ABSTRACT

The design of an optical computer must be based on the characteristics of optics and optical technology, and not on those of electronic technology. In this paper the property of optical superposition is considered and the implications it has in the design of computing systems is discussed. It can be exploited in the implementation of optical gates, interconnections, and shared memory.

INTRODUCTION

Fundamental differences in the properties of electrons and photons provide for expected differences in computational systems based on these elements. Some, such as the relative ease with which optics can implement regular, massively parallel interconnections are well known. In this paper we examine how the property of superposition of optical signals in a linear medium can be exploited in building an optical or hybrid optical/electronic computer. This property enables many optical signals to pass through the same point in space at the same time without causing mutual interference or crosstalk. Since electrons do not have this property, this may shed more light on the role that optics could play in computing. We will separately consider the use of this property in interconnections, gates, and memory.

INTERCONNECTIONS

A technique for implementing optical interconnections from one 2-D array to another (or within the same array) has been described [Jenkins et al, 1984]. It utilizes two holograms in succession (Fig. 1). The holograms

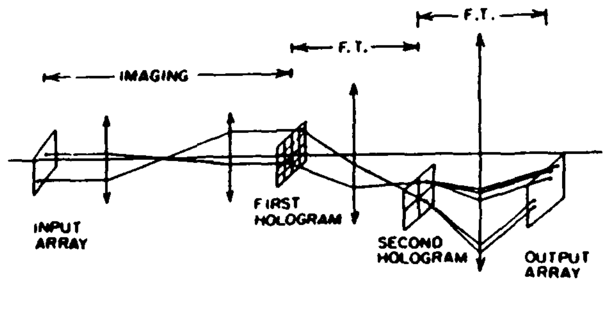


Fig. 1. Optical holographic system for interconnections.

can be generated by a computer plotting device. The idea is to define a finite number, M, of distinct interconnection patterns, and then assemble the interconnecting network using only these M patterns. The second hologram of Fig. 1 consists of an array of facets, one for each of the M interconnection patterns. The first hologram contains one facet for each input node, and serves to address the appropriate patterns in the second hologram.

It is the superposition property that makes this interesting. Note that many different signal beams can pass through the same facet of the second hologram at the same time without causing mutual interference. (All of these signals merely get shifted in the same direction and by the same amount.) This feature decreases the complexity of both holograms -- The first because it only has to address M facets, the second hologram because it only has M facets. Let N be the number of nodes in the input and output arrays. The complexity (number of resolvable spots) of each hologram can be shown to be proportional to NM, with the proportionality constant being approximately 25 [Jenkins et al., 1984].

Using this as a model for interconnections in parallel computing, a comparison can be made between the complexity of these optical interconnections with those of electronic VLSI for various interconnection networks. Results of this have been given in [Giles and Jenkins, 1986]. It is found that in general the optical interconnections have an equal or lower space complexity than electronic interconnections, with the difference becoming more pronounced as the connectivity increases.

SHARED MEMORY

The same superposition principle can be applied to memory cells, where many optical beams can read the same memory location simultaneously. This concept could be useful in building a parallel shared memory machine.

For this concept, we first consider abstract models of parallel computation based on shared memories. The reason for this approach is to abstract out inherent limitations of electronic technology (such as limited interconnection capability); in designing an architecture one would adapt the abstract model to the limitations of optical systems. These shared memory models are basically a parallelization of the Random Access Machine.

The Random Access Machine (RAM) model [Aho, Hopcroft, and Ullman, 1974] is a model of sequential computation, similar to but less primitive than the Turing machine. The RAM model is a one-accumulator computer in which the instructions are not allowed to modify themselves. A RAM consists of a read-only input tape, a write-only output tape, a program and a memory. The time on the RAM is bounded above by a polynomial function of time on the TM. The program of a RAM is not stored in memory and is unmodifiable. The RAM instruction set is is small and consists of operations such as store, add, subtract, and jump if greater than zero; indirect addresses are permitted. A common RAM model is the uniform cost one, which assumes that each RAM instruction requires one unit of time and each register one unit of space.

Shared memory models are based on global memories and are differentiated by their accessibility to memory. In Fig. 2 we see a typical shared memory

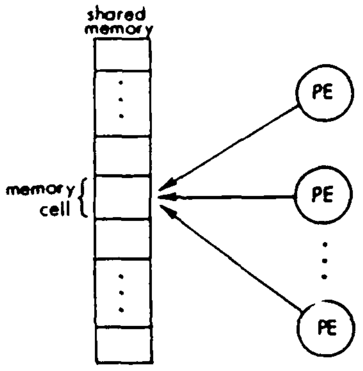


Fig. 2. Conceptual diagram of shared memory models.

model where individual processing elements (PE's) have variable simultaneous access to an individual memory cell. Each PE can access any cell of the global memory in unit time. In addition, many PE's can access many different cells of the global memory simultaneously. In the models we discuss, each PE is a slightly modified RAM without the input and output tapes, and with a modified instruction set to permit access to the global memory. A separate input for the machine is provided. A given processor can generally not access the local memory of other processors.

The various shared memory models differ primarily in whether they allow simultaneous reads and/or writes to the *same* memory cell. The PRAC, parallel random access computer (Lev, Pippenger and Valiant, 1981) does not allow simultaneous reading or writing to an individual memory cell. The PRAM, parallel random access machine, (Fortune and Wyllie, 1978) permits simultaneous reads but not simultaneous writes to an individual memory cell. The WRAM, parallel write random access machine, denotes a variety of models that permit simultaneous reads and certain writes, but differ in how the write conflicts are resolved. For example, a model by Shiloach and Vishkin (1981) allows a simultaneous write only if all processors are trying to write the same value. The paracomputer (Schwartz, 1980) has simultaneous writes but only "some" of all the information written to the cell is recorded. The models represent a hierarchy of time complexity given by

$$T^{\mathrm{PRAC}} \geq T^{\mathrm{PRAM}} \geq T^{\mathrm{WRAM}}$$

where T is the minimum number of parallel time steps required to execute an algorithm on each model. More detailed comparisons are dependent on the algorithm (Borodin and Hopcroft, 1985).

In general, none of these shared memory are physically realizable because of actual fan-in limitations. As an electronic example, the ultracomputer (Schwartz, 1980) is an architectural manifestation of the paracomputer that uses a hardwired Omega network between the PE's and memories; it simulates the paracomputer within a time penalty of $O(\log^2 n)$.

Optical systems could in principle be used to implement this parallel memory read capability. As a simple example, a single 1-bit memory cell can be represented by one pixel of a 1-D or 2-D array; the bit could be represented by the state (opaque or transparent) of the memory cell. Many optical beams can simultaneously read the contents of this memory cell without contention (Fig. 3). In addition to this an interconnection network

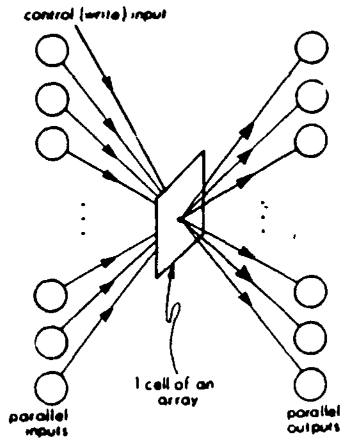


Fig. 3. One memory cell of an array, showing multiple optical beams providing contention-free read access.

is needed between the PE's and the memory, that can allow any PE to communicate with any memory cell, preferably in one step, and with no contention. A regular crossbar is not sufficient for this because fan-in to a given memory cell must be allowed. Figure 4 shows a conceptual block diagram of a system based on the PRAM model; here the memory array operates in reflection instead of transmission. The fan-in required of the interconnection network is also depicted in the figure.

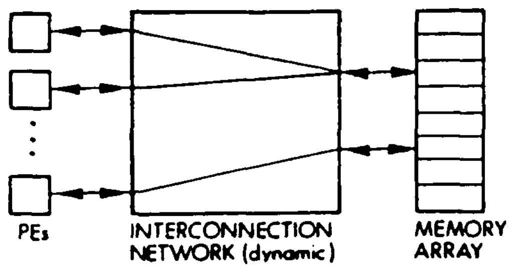


Fig. 4. Block diagram of an optical architecture based on parallel RAM models.

Optical systems can potentially implement crossbars that also allow this fan-in. Several optical crossbar designs discussed in (Sawchuk, et al., 1986) exhibit fan-in capability. An example is the optical crossbar shown schematically in Fig. 5. The 1-D array on the left could be optical sources (LED's or laser diodes) or just the location of optical signals entering from previous components. An optical system spreads the light from each input source into a vertical column that illuminates the crossbar mask. Following the crossbar mask, a set of optics collects the light transmitted by each row of the mask onto one element of the output array. The states of the pixels in the crossbar mask (transparent or

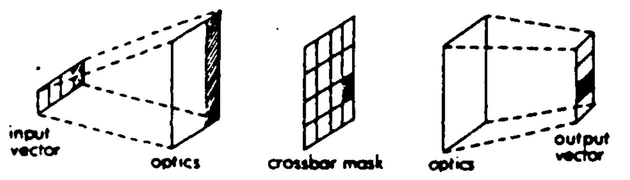


Fig. 5. Example of an optical crossbar interconnection network.

opaque) determine the state of the crossbar switch. Multiple transparent pixels in a column provide fanout; multiple transparent pixels in a row provide fan-in. Many optical reconfigurable network designs are possible, and provide tradeoffs in performance parameters such as bandwidth, reconfiguration time, maximum number of lines, hardware requirements, etc. Unfortunately, most simple optical crossbars will be limited in size to approximately 256 x 256 (Sawchuk, et al., 1986). We are currently considering variants of this technique to increase the number of elements. Possibilities include using a multistage but nonblocking interconnection network (e.g. Clos), a hierarchy of crossbars, and/or a memory hierarchy.

GATES

Since the superposition property of optics only applies in linear media, it cannot in general be used for gates, which of course are inherently nonlinear. However, for important special cases superposition can allow many optical gates to be replaced with one optical switch.

Consider again the situation depicted in Fig. 3, with the aperture being used as a switch or relay. The control beam opens or closes the relay; when the relay is closed (i.e., aperture is transparent), many optical signal beams can independently pass through the relay. If b represents the control beam and a_i the signal beams, this in effect computes $b \cdot a_i$ or $\bar{b} \cdot a_i$, depending on which state of b closes the relay, where $\cdot$ denotes the AND operation (Fig. 6).

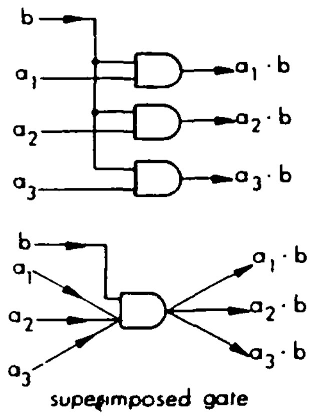


Fig. 6. One optical relay or superimposed gate versus individual gates with a common input.

Using this concept, a set of gates with a common input in an SIMD machine can be replaced with one optical switch or "superimposed gate". It also obviates the need for broadcasting the instructions to all PE's; instead, a fan-in of all signals to a common control switch is performed.

These superimposed gates are not true 3-terminal devices. The common (b) input is regenerated, but the a_i inputs are not. As a result, a design constraint, that these a_i signals do not go through too many superimposed gates in succession without being regenerated by a conventional gate, must be adhered to. Another consequence is that the total switching energy required for a given processing operation is reduced, because N gates are replaced with one superimposed gate. This is important because it is likely that the total switching energy will ultimately be the major limiting factor on the switching speed and number of gates in an optical computer. Other advantages include an increase in computing speed since some of the gates are effectively passive and reduced requirements on the device used to implement the optical gates.

CONCLUSIONS

We have shown that the property of superposition can be exploited in the design of optical or hybrid optical/electronic computing architectures. It can reduce the hologram complexity for highly parallel interconnections, reduce the number of gates in a SIMD system, and permit simultaneous memory access in a parallel shared memory machine, thereby reducing contention problems. Our fundamental premise in studying this is that architectures for optical computing must be designed for the capabilities and limitations of optics; they must not be constrained by the limitations of electronic systems, which have necessarily dominated approaches to digital parallel computing architectures to date.

REFERENCES

Aho, A.V., J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, Mass. Addison-Wesley, 1974.

Borodin. A. and J.E. Hopcroft, "Routing, Merging, and Sorting on Parallel Models of Computation," *Journal of Computer and System Sciences*, Vol. 30, pp. 130-145 1985.

Fortune. S., and J. Wyllie, "Parallelism in Random Access Machines," *Proc 10th Annual ACM STOC*, San Diego, California, pp. 114-118, 1978.

Giles. C. L., and Jenkins, B. K., "Complexity Implications of Optical Parallel Computing," *Proc. Twentieth Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, Calif., to appear, Nov. 1986.

Jenkins, B.K., et al., "Architectural Implications of a Digital Optical Processor," *Applied Optics*, Vol. 23, No. 19, pp. 3465-3474, 1984.

Lev, G., N. Pippenger, and L.G. Valiant, "A Fast Parallel Algorithm for Routing in Permutation Networks" *IEEE Trans. on Computers*, Vol. C-30, No. 2, pp. 33-100, Feb 1981.

Sawchuk, A. A., B.K. Jenkins, C.S. Raghavendra, and A. Varma, "Optical Matrix-Vector Implementations of Crossbar Interconnection Networks," *Proc. International Conference on Parallel Processing*, St. Charles, IL, August 1986; also Sawchuk, et al., "Optical Interconnection Networks," *Proc. International Conference on Parallel Processing*, pp. 388-392, August 1985.

Schwartz, J.T., "Ultracomputers," *A.C.M. Trans on Prog. Lang. and Sys.*, Vol. 2, No. 4, pp. 484-521, October 1980.

Shiloach, Y., and Vishkin, U., "Finding the Maximum, Merging and Sorting in a Parallel Computation Model," *J. Algorithms*, pp. 88-102, March 1981.